

**CERIAS Tech Report 2005-128**

**VIDEO AND IMAGE WATERMARK SYNCHRONIZATION**

by Eugene Ted Lin

Center for Education and Research in  
Information Assurance and Security,  
Purdue University, West Lafayette, IN 47907-2086

VIDEO AND IMAGE WATERMARK SYNCHRONIZATION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Eugene Ted Lin

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2005

To My Parents and Brother

## ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Edward J. Delp, for his helpful advice, invaluable guidance, and for “keeping my mind right.” He has provided many opportunities to explore the most interesting and challenging pursuits, and his teachings in scholarly research and in life shall always be cherished and remembered. He has been an advisor of the highest regard.

I would also like to thank my graduate committee: Professors Leah H. Jamieson, Mary P. Harper, and Zygmunt Pizlo, for their advice, encouragement, criticism, and insight. They have encouraged me to look deeper, to explore and discover new perspectives.

I appreciate the support and friendship of all my colleagues in the Video and Image Processing (VIPER) lab, past and present: Dr. Gregory Cook, Dr. Cuneyt Taskiran, Dr. Paul Salama, Dr. Eduardo Asbun, Dr. Sheng Liu, Dr. Yuxin Liu (Zoe), Dr. Jinwha Yang, Dr. Sahng-Gyu Park, Dr. Yajie Sun, Dr. Lauren Christopher, Anthony Martone, Aravind Mikkilineni, Hyung Cook Kim, Hwayoung Um, Zhen Li, Limin Liu, Jennifer Talavage, Michael Igarta, Oriol Guitart Pla, Hakeem Ogunleye, Martha Saenz, Dan Hintz, and Ray Wolfgang. This journey would not have been as special, or as memorable, without their camaraderie and friendship. I would like to thank my colleagues from abroad for their encouragement and perspective: Professor Luis Torres, Dr. Josep Prades-Nebot, Dr. Alberto Albiol, Rafael Villoria, and Andreas Lang. I would also like to thank Dan Teany and Mercan Topkara for their support, advice, and many enjoyable discussions. Dan was involved in so many occasions that he was practically a member of our lab.

I would especially like to thank Cuneyt “Taskmaster” Taskiran for providing his unique outlook on all things; Jinwha “Texcan” Yang for providing so much feedback and advice; Gregory Cook for all his knowledge of “how things work”;

Yuxin Liu for being an inspiration to work even harder and for the “real” perspective about China; Anthony “TONLOC” Martone for bringing out the lighter side of our advisor; Aravind “Qrovna ho3e-y337: Cja3q Zvpebfbg” Mikkilineni and Dan “User-Thread” Teany for too many geeky conversations about Linux, C++, and anime; Professor Thomas “=/Knife-” Talavage and Jennifer “Shira” Talavage for luring the geeks out of the lab to appreciate life; Michael “bling-bling” Igarta for one more “secret” project; and Rafael “Depredator” Villoria for professing the finest qualities of American gourmet and movies.

I would also like to thank Professor Ahmet Eskicioglu, Professor Reginald Lagendijk, Professor Jana Dittmann, Dr. Ton Kalker, Dr. Christine Podilchuk, Dr. Adnan Alattar, and Dr. Mehmet Celik, with whom I have had the pleasure of collaborating.

Lastly, I would like to thank my parents for unwavering support and encouragement. My father has encouraged me to try my best, work hard at every endeavor, and persevere because one does not find the rewards in life handed upon silver platters. My mother has taken care of “all the little things” that I have neglected over these many years, put up with me coming home from the lab at 4:30 am, and taught patience and compassion. Finally, I would like to express my deepest gratitude to Jeff, for being the best brother in the world.

## TABLE OF CONTENTS

	Page
LIST OF TABLES . . . . .	viii
LIST OF FIGURES . . . . .	ix
ABSTRACT . . . . .	xiii
1 INTRODUCTION . . . . .	1
1.1 Content Protection . . . . .	2
1.2 Encryption . . . . .	4
1.3 Watermarking . . . . .	5
1.3.1 Watermarking applications . . . . .	6
1.3.2 Synchronization and synchronization attacks . . . . .	8
1.4 Overview of the Dissertation . . . . .	9
2 BACKGROUND . . . . .	11
2.1 Watermarking Overview . . . . .	11
2.1.1 Watermark embedding . . . . .	12
2.1.2 Watermark attack . . . . .	16
2.1.3 Watermark detection . . . . .	21
2.2 Watermarking Objectives and Evaluation . . . . .	27
2.2.1 Perceptual transparency . . . . .	27
2.2.2 Robustness . . . . .	30
2.2.3 Capacity . . . . .	31
2.2.4 Security . . . . .	32
2.2.5 Other criteria . . . . .	33
2.2.6 Performance evaluation, benchmarking, and tradeoffs . . . . .	34
2.3 Digital Images and Video . . . . .	36
2.4 Digital Video Compression . . . . .	40

	Page
2.4.1	Video compression standards . . . . . 41
2.4.2	Hybrid predictive-transform coding . . . . . 45
2.5	Image and Video Watermarking . . . . . 50
2.5.1	Image watermarking . . . . . 50
2.5.2	Video watermarking . . . . . 53
2.6	Synchronization . . . . . 61
2.6.1	Synchronization attacks . . . . . 65
2.6.2	Synchronization insensitivity and invariant transformations . . 69
2.6.3	Synchronization and templates . . . . . 72
2.7	Additional Background . . . . . 77
2.7.1	Pseudo-random number generators . . . . . 77
2.7.2	State machines . . . . . 79
2.7.3	Steganography . . . . . 81
3	TEMPORAL SYNCHRONIZATION . . . . . 83
3.1	Temporal Synchronization Framework . . . . . 83
3.2	Watermark Embedding Model . . . . . 88
3.3	Watermark Detection Model . . . . . 92
3.4	Analysis . . . . . 98
3.5	Temporal Redundancy and Synchronization . . . . . 104
3.5.1	Adaptive state transitions . . . . . 111
3.5.2	Security . . . . . 113
3.6	Experimental Results . . . . . 115
3.6.1	Frame averaging . . . . . 127
3.6.2	Adaptive state transitions . . . . . 131
3.7	Conclusions and Future Work . . . . . 135
4	SPATIAL SYNCHRONIZATION . . . . . 139
4.1	Watermark Embedding . . . . . 140
4.2	Watermark Synchronization . . . . . 143

	Page
4.2.1 Watermark scale estimation . . . . .	147
4.2.2 Watermark rotation estimation . . . . .	154
4.3 Experimental Results . . . . .	155
4.3.1 Scale and rotation rank . . . . .	156
4.3.2 Discussion . . . . .	157
4.4 Conclusions and Future Work . . . . .	169
5 SEMI-FRAGILE WATERMARKING . . . . .	171
5.1 Robust, Fragile, and Semi-Fragile Watermarking . . . . .	171
5.1.1 Semi-fragile watermarking . . . . .	174
5.2 A Semi-Fragile Watermark . . . . .	176
5.2.1 Watermark generation and insertion . . . . .	176
5.2.2 Watermark Detection . . . . .	178
5.3 Evaluation of the Semi-Fragile Watermark . . . . .	180
5.4 Conclusions . . . . .	188
6 CONCLUSIONS . . . . .	190
6.1 Contributions . . . . .	191
6.2 Future Work . . . . .	192
LIST OF REFERENCES . . . . .	195
VITA . . . . .	218



## LIST OF TABLES

Table	Page
2.1	Desired output of ideal robust watermark detector . . . . . 23
2.2	Observation of the watermarked signal $\hat{Y}$ by users and watermark detectors. The user is interested in the original signal. Watermark detectors are interested in the watermark. A non-blind detector has access to the original signal and can remove host-signal interference. . . . . 25
2.3	Common sizes of digital video frames . . . . . 39
2.4	Examples of spatial synchronization attacks . . . . . 67
3.1	Structure of watermarked video using adaptive state transitions for <i>Akiyo</i> 132
3.2	Structure of watermarked video using adaptive state transitions for <i>Foreman</i> 133
3.3	Structure of watermarked video using adaptive state transitions for <i>Bus</i> . 133
3.4	Structure of watermarked video using adaptive state transitions (Mean over all videos) . . . . . 134
5.1	Block Statistics for detector (embedding strength $\gamma = 5.0$ , detection blocksize= $16 \times 16$ ) . . . . . 182

## LIST OF FIGURES

Figure	Page
2.1 Classical model of watermarking . . . . .	12
2.2 Watermark embedding . . . . .	14
2.3 Original image <i>Fruit</i> . . . . .	16
2.4 Watermarked <i>Fruit</i> and difference images using technique of [61]. Distortion introduced by watermark insertion is more visible as the strength $\gamma$ is increased. . . . .	17
2.5 Mosaic Attack on the watermarked <i>Fruit</i> image. Each block represents an individual image. The gaps between the images are not present when the images are displayed. Watermark detection fails when each of the small images is examined individually. . . . .	19
2.6 Visibility of distortion in <i>Flowergarden</i> . Additive white noise has been inserted into the image. The noise is independent of the image and uniformly distributed with zero mean. The PSNR of the noisy image is 27.67 dB. The distortion is not very noticeable at the bottom (textured flower region) of the noisy image but obvious at the top (smooth sky region). . . . .	29
2.7 Chrominance subsampling showing positions of chrominance pixels relative to luminance pixels ( $\times$ =Luminance pixel, $\circ$ =Chrominance pixel) . . . . .	38
2.8 Video compression . . . . .	42
2.9 Example of the object-oriented video compression supported by MPEG-4. Each video object is encoded as a separate elementary stream. The elementary streams for all the objects are multiplexed within the MPEG-4 compressed stream. An MPEG-4 decoder composes the scene from the individual objects for display. . . . .	46
2.10 Hybrid predictive-transform coding framework. The encoder contains a fully functional decoder within the prediction loop. . . . .	47
2.11 Example of temporal prediction used in hybrid coding. I-frames are white, P-frames are black, and B-frames are gray. Frames are numbered to indicate display order. Arrows indicate temporal prediction, pointing from a reference frame to the predicted inter-frame. . . . .	48

Figure	Page
2.12 Compressed domain watermark embedding. The original compressed video is parsed and partially decoded to expose elements of the compressed video data. Some of the elements are modified to insert the watermark, and then the compressed data is reassembled to obtain the watermarked video. . . . .	57
2.13 Watermark embedding, detection and synchronization . . . . .	62
2.14 Watermark detection with synchronization . . . . .	73
3.1 Watermark embedding model . . . . .	89
3.2 Example key generators . . . . .	93
3.3 Watermark detection model . . . . .	94
3.4 State sequence of modified embedder. Each circle represents the state used to watermark a single video frame. Arrows indicate frames where state transitions occur in the key generator. . . . .	106
3.5 State sequence in a frame dropping attack . . . . .	108
3.6 Modified embedder with temporal redundancy control . . . . .	110
3.7 Example of changing feature values reducing temporal redundancy . . . . .	112
3.8 Detection rate under frame dropping attack . . . . .	121
3.9 Detection rate under frame decimation attack . . . . .	122
3.10 Detection rate under frame decimation attack (continued) . . . . .	123
3.11 Detection rate under frame transposition attack . . . . .	124
3.12 Detection rate under frame averaging attack . . . . .	125
3.13 Detection rate under frame averaging attack (continued) . . . . .	126
3.14 Frame averaging attack on watermarked video ( $\beta = 5$ ) with window size $\lambda = 3$ . Each box indicates a separate frame of the watermarked video, with the watermarks embedded in each frame indicated. . . . .	130
4.1 Watermark embedder with state machine key generator for spatial synchronization . . . . .	141
4.2 Watermark structure showing macroblocks and blocks. The image shows an example of the structure of a watermark constructed and inserted in the spatial domain. The first block of all macroblocks are identical. . . . .	142

Figure	Page
4.3 Autocorrelation of watermark showing location of local maxima or peaks. Peaks are indicated by $\times$ , forming a regular grid. Not all the peaks in the autocorrelation are shown. . . . .	145
4.4 Spatial synchronization procedure . . . . .	146
4.5 Example of watermark scale and rotation estimation using image <i>Girl</i> . PSNR of watermarked image is 33.8 dB, $M = 3$ , $B = 16$ . Attack is re-scaling by factor $f = 1.15$ followed by rotation of $6^\circ$ . Neighbor distance is $\chi = MBf = 55.2$ pixels. . . . .	148
4.6 Distance histogram of <i>Girl</i> (histogram bin size $\Delta_s = 1$ ). Large values occur at the bins corresponding to the distances of 55 pixels, 78 pixels, 110 pixels, and 123 pixels. . . . .	149
4.7 Improving the scale estimation . . . . .	151
4.8 Score function $S_s(k\Delta_s)$ of <i>Girl</i> ( $\Delta_s = 1$ ). The largest score occurs for the distance of 55 pixels, which implies that scale estimation is correct on the first attempt. . . . .	153
4.9 Original and watermarked <i>Girl</i> and <i>Crowd</i> ( $\gamma = 5.0$ , $M = 2$ , $B = 16$ ) . .	162
4.10 Original and watermarked <i>Fruit</i> and <i>Peppers</i> ( $\gamma = 5.0$ , $M = 2$ , $B = 16$ ) .	163
4.11 Scale and rotate ranks for scaling attack (30% to 210%) without rotation, with watermark construction parameters $M$ and $B$ varied. . . . .	164
4.12 Scale and rotate ranks for rotation attack ( $0^\circ$ to $89^\circ$ ) without scaling, with watermark construction parameters $M$ and $B$ varied. . . . .	165
4.13 Scale and rotate ranks for rotation and scaling attack. Scaling is varied from 30% to 210% while rotation is fixed at $3^\circ$ . Watermark construction parameters $M$ and $B$ varied. . . . .	166
4.14 Scale and rotate ranks for rotation and scaling attack. Scaling is fixed at 120% while rotation is varied from $0^\circ$ to $89^\circ$ . Watermark construction parameters $M$ and $B$ varied. . . . .	167
4.15 Effect of image padding. When an image is rotated near $45^\circ$ , a significant area of the attacked image is occupied by padding (the black triangular regions.) These regions, and the strong edges between the padded regions and the image, disrupt the pattern of peaks in the autocorrelation. . . .	168
5.1 Watermark generation in DCT domain for $8 \times 8$ blocks. For each watermark block, white coefficients are generated using a PRNG. Shaded coefficients are not watermarked and have a value of zero. . . . .	177
5.2 Example of block operators on a block $B$ of size $4 \times 4$ samples . . . . .	178

Figure	Page
5.3 Original Images . . . . .	183
5.4 Altered Images . . . . .	184
5.5 Mean correlation of unaltered and altered blocks when the tampered image is provided to the watermark detector (embedding strength $\gamma = 5.0$ , blocksize= $16 \times 16$ ) after varying degrees of JPEG compression. The accuracy of the detector improves when there is a large separation between the mean $\rho_b$ of altered and unaltered blocks. . . . .	185
5.6 Detector accuracy when the threshold is varied from 0.0 to 1.0, for a tampered image compressed at various JPEG quality factors. When the threshold is near 0.0, almost all of the incorrect detections are misses. When the threshold is near 1.0, almost all of the incorrect detections are false positives. . . . .	186
5.7 Example of Detection. $\gamma = 5.0$ , $T = 0.1$ , blocksize= $16 \times 16$ , JPEG $Q = 60$ data rate=0.90 bits/pixel, 93% correct detection (5332 blocks correct out of 5704 blocks), 4% false positive (211 false positives out of 4753 unaltered blocks), 17% misses (161 misses out of 951 altered blocks). A box indicates an altered block correctly identified, X indicates false positive, and X within a box indicates a miss. . . . .	187

## ABSTRACT

Lin, Eugene T. Ph.D., Purdue University, May, 2005. Video and Image Watermark Synchronization. Major Professor: Edward J. Delp.

Digital watermarking is the practice of inserting a signal, known as the watermark, into an original signal in an imperceptible manner. The watermark encodes or represents information that can protect the watermarked signal, typically identifying the owner (source) or the intended recipient (destination) of the signal. The embedded watermark may be detected by using a watermark detector, which enables an application to react to the presence (or absence) of the watermark in a signal. However, the watermarked signal may be processed, or attacked, prior to watermark detection. Attacks may remove the embedded watermark or make the watermark more difficult to detect.

One type of attack that has received considerable attention is synchronization attacks. A synchronization attack confuses the watermark detector by re-positioning the embedded watermark. Most watermark detectors will fail to detect the watermark embedded in the attacked signal unless the position of the watermark can be identified. This is a significant vulnerability in robust watermark detection. The process of identifying the position of the watermark is known as watermark detector synchronization.

A new framework is developed for temporal synchronization in blind symmetric video watermarking. Embedding and detection models are proposed that encompass the behavior of many video watermarking techniques. These models demonstrate that synchronization is challenging when the watermark lacks redundancy, but also that efficient synchronization can be achieved by designing the watermark with temporal redundancy. The temporal synchronization models are adapted to

spatial synchronization in still image watermarks. For spatial synchronization, redundancy is obtained by constructing a watermark which induces a pattern in the auto-correlation. Experimental results support the theoretical foundations for both temporal and spatial synchronization.

In addition, earlier exploration in watermarking led to the development of a semi-fragile watermarking technique for image authentication. The semi-fragile technique is capable of detecting significant alterations to the watermarked image, but is tolerant to lossy JPEG compression and other, more subtle alterations. This earlier work is not related to watermark synchronization.

## 1. INTRODUCTION

The use of digital video has grown dramatically in recent times. Digital video applications include video-conferencing, video-on-demand, digital television, digital cinema, distance learning, entertainment, surveillance, and advertising. Many users experience digital video when they watch a motion picture recorded on a digital video disc (DVD) or downloaded over the Internet. The proliferation of digital video into more applications is encouraged by improving compression technology, better authoring and editing tools, lower-cost capture and display devices, and more available bandwidth in digital communication networks.

Digital representation offers many advantages for processing and distributing video and other types of information. First, digital software programs offer unprecedented flexibility in creating, editing, presenting, and manipulating the digital information. Analog devices lack the flexibility, malleability, and extensibility of software processing. Second, digital communications networks (such as the Internet) allow digital data to be distributed and disseminated on a wide scale. On some of these networks, existing open and proprietary protocols such as the World Wide Web allow any user to easily and inexpensively obtain, provide, exchange, and find digital information. Lastly, digital information can be processed, and in particular, copied without introducing loss, degradation, or noise [1]. For example, an unlimited number of perfect copies can be produced from a single digital video signal. In contrast, the addition of noise into a copy from analog signal processing is unavoidable. (That is, a copy of an analog video signal is not an exact replica of the original.)

While the aforementioned advantages offer immense opportunities for creators, the ability to make perfect copies and the ease by which those copies can be distributed also facilitate misuse, illegal copying and distribution (“piracy”), plagiarism,



and misappropriation. Content creators and owners are concerned about the consequences of illegal copying and distribution on a massive scale. This problem is not merely theoretical. The economic damage arising from illegal copying and distribution of copyrighted materials is estimated to be in the billions of dollars [2, 3]. Recently, popular Internet software based on a peer-to-peer (P2P) architecture (such as *Kazaa* [4], *BitTorrent* [5], *eDonkey* [6] and *Gnutella*) has been used to share (distribute) copyrighted music, movies, software, and other materials. Future P2P systems may encrypt the data being shared, preserve the anonymity of its users, support a larger number of users, and be more robust [7, 8]. These advances in P2P systems will create considerable challenges for copyright enforcement. Thus, there is a great desire for methods which can preserve the economic value of digital video and protect the rights of content owners.

## 1.1 Content Protection

Content protection [2] is a challenging problem which involves conflicting interests. Content owners wish to ensure that their intellectual property is not misused, illegally copied, or distributed. Device manufacturers desire to keep their products inexpensive and simple, however, implementing technological content protection measures increases both the cost and complexity of devices. Device manufacturers also desire to minimize the risk of introducing devices to market which restrict or limit the activities of their users, as they are aware that potential customers (greatly) shun such devices. Users desire their legal privileges (such as First Sale and Fair Use) [9–11] and privacy [12] to be safeguarded. Users do not wish to pay for devices which are more expensive, more complex, less compatible (interoperable), and more restrictive of their activities. Users are wary that they shall burden the costs for content protection systems, even though the primary beneficiaries of such systems are content owners and providers.

While it is not clear where the balancing point amongst the diverse interests ultimately lie, there has been technical and legal initiatives towards content protection [2]. The Digital Millennium Copyright Act (DMCA) was the first legislation in a series of efforts by the U.S. Congress to update the U.S. copyright law for the digital age. President Clinton signed the Act into law on October 28, 1998. The DMCA prohibit circumvention of technological measures used by copyright owners to protect their works.<sup>1</sup> Similar provisions appear in the European Copyright Directive (EUCD), or Directive 2001/29/EC of the European parliament. The EUCD obliges the Member States to call into being legal protection against the circumvention of technical security measures as well as against manufacturing, offering for sale or trading in equipment the primary object of which is to circumvent these technical security measures. Member States of the European Union are currently implementing the directive, be it with a much slower pace than what the implementation deadline of December 2002 called for. Furthermore, some national parliaments have rejected (initial) proposals to implement the EUCD.

To complement the legal initiatives, content owners have also sought technical measures to protect their works. Technical content protection measures generally use three approaches: access control, copy protection, and content tracking.

The objective of access control is to ensure that video is accessible only under the rules or conditions specified by the content owner. For example, the owner may specify that only certain users may view the video, or that the video can be viewed for a limited number of times, or that payment is required each time the video is viewed. Access control alone offers limited protection, however, because at some point the video will be provided to a user. When this occurs, access control will not prevent the user from creating illicit copies or misusing the video.

Copy protection [14] prevents or hinders the creation of copies. Copies of a video can be created digitally, or by recording the video as it is displayed to the user.

---

<sup>1</sup>These provisions are controversial. In particular, it may be illegal to circumvent technical protection measures even if the intended reason for doing so falls under Fair Use provisions or if making a copy is otherwise legal [11, 13].

Recording the video as it is displayed for the intent of circumventing or removing technical content protection measures is known as exploiting the “analog hole” [15]. A copy protection system identifies copy protected video and then uses some means to prevent or make more difficult the creation of a copy. Unfortunately, copy protection is very challenging and techniques for copy protection have consistently been defeated.

Content tracking is a means to protect video even if access control and copy protection are circumvented. In content tracking, each legitimate copy of the video is personalized or individualized by embedding information indicating the user who has custody of that copy. If a copy of the video is discovered in a suspicious location (such as being shared using P2P software), an investigator may extract the embedded information from the copy and determine source or origin of the suspect copy. The content owner may decide to initiate action if many suspect copies are found.

## 1.2 Encryption

Access control has often been addressed by using encryption. Encryption [16–25] is the process of scrambling data into an unintelligible form. The original data is known as the plaintext and the scrambled data is known as the ciphertext. The inverse process of obtaining the plaintext from the ciphertext is known as decryption. Encryption provides confidentiality because a secret key is necessary for decryption. Traditionally, encryption has been used to ensure the confidentiality of sensitive information (such as electronic mail, military secrets, and financial information) transmitted through an insecure communications channel. For access control, video is encrypted and the decryption key is provided only after the access conditions have been satisfied. Obtaining the encrypted video alone (without the decryption key) does not allow the video to be displayed.

The significant limitation of encryption is that it offers no protection once the video is decrypted. This implies that encryption alone is not sufficient for content

protection and another method is needed to protect the video after decryption. Watermarking has been proposed to provide more lasting protection.

### 1.3 Watermarking

Digital watermarking [26–33] is the insertion of a signal, known as the watermark, into original video in an imperceptible manner. The watermark encodes or represents information that can protect the video, typically identifying the owner (source) or the intended recipient (destination) of the video. The process of inserting the watermark introduces distortion, however, watermarking techniques use heuristics or perceptual models [32] to conceal the presence of the watermark embedded in the watermarked video. Ideally, the watermarked and original videos are perceptually indistinguishable when displayed. The embedded watermark may be detected by using a watermark detector, which enables an application to react to the presence (or absence) of the watermark in a video. In addition to video, watermarking techniques have been proposed to protect images, audio, text, and other types of data [29].

A challenge in watermarking is that processing the watermarked video may remove or damage the embedded watermark, or make the watermark more difficult to detect. The watermarked video may be processed for any number of reasons, including the normal processing that occurs in an application; unintentional damage or loss during storage, retrieval, or transmission over a network; or deliberate processing by a (hostile) user for the purpose of removing the embedded watermark. Processing the watermarked video is known as an attack, whether the intent such processing is malicious or not.

In addition to perceptually transparent embedding, another goal of watermarking is robustness against attacks.<sup>2</sup> The watermark detector should detect the watermark in the watermarked video, even when the video has been subjected to attack. The

---

<sup>2</sup>Robustness is not a desired property in some watermarks used for authentication. This will be discussed in Chapter 5.

watermark should be securely embedded and difficult to remove, such that the embedded watermark is a permanent and inseparable part of the watermarked video.

### 1.3.1 Watermarking applications

While content protection (including content tracking) has often been mentioned as the motivation for digital watermarking, watermarking can be used in other applications as well. Applications and potential applications of watermarking include [2, 28, 34–37]:

- **Content Tracking:** The owner personalizes each copy of the content by embedding a watermark into the copy. The embedded watermark identifies the user who has custody of that copy. Any subsequent digital copies made of the watermarked content will also be watermarked. If a suspicious copy of the content is discovered, detecting the watermark reveals the source of the suspect copy. These watermarks are sometimes referred to as fingerprints.

Content tracking is not necessarily directed at individual users. For example, consider the mass production of pre-recorded video. Suppose the video owner contracts the services of various mastering and distribution companies to create and distribute the video on media. However, the owner is worried that some companies may have insufficient security measures to safeguard the video. Unscrupulous companies or employees may even conspire to “leak” illicit copies to pirates. To trace security breaches, the owner embeds a different watermark into the copies he provides to each mastering company. If illicit copies are found before the official release of the video, the video owner detects the watermark to identify the company whose security is lacking. The content owner may then choose not to deal with that company in the future. A similar application lies in digital cinema, where the movie owner is worried about collusion between some theater owners and pirates.

- **Owner or Copyright Identification:** In copyright watermarking, the embedded watermark encodes ownership information such as the identity of the owner and the copyright date. Detecting the watermark provides the content owner with additional evidence of ownership in the event of a dispute. The embedded information may also be useful in detecting or showing plagiarism, particularly when the watermark is detected in allegedly original content from a third party.
- **Copy Protection:** The presence of the watermark identifies the watermarked content as copy protected. A device that obeys the copy protection protocol detects the watermark and then disallows the creation of copies. Some copy protection schemes allow the user to create a single generational copy but restrict the user from making additional copies from a copy [14]. In such schemes, an embedded watermark could encode information such as “always allow additional copies”, “only one additional copy allowed”, and “no more copies allowed”. Using watermarks in this manner requires cooperation from recording devices to detect the watermark and prevent unauthorized copying. The embedded watermark will not prevent the video from being copied if the recording device does not detect or ignores the watermark.
- **Broadcast Monitoring [38]:** An embedded watermark may be used to recognize or identify a signal of interest, particularly when the signal has been spliced or merged with other signals. Recognition occurs when the watermark is detected. For example, an advertiser wishes to verify that a particular advertisement is being broadcast as contracted. Verification and auditing are important considerations when the production and distribution of broadcast video content, including advertisements, entertainment content, and news, have immense economic value.
- **Authentication:** The ability to detect altered or forged video is critical in applications such as video surveillance [39,40]. An embedded watermark encodes information necessary to verify the integrity of the watermarked signal [41–44].

If alterations are detected, the watermark allows the identification of the altered regions (though not necessarily the nature of the alterations). Authentication also encompasses anti-counterfeiting, where a watermark is embedded to increase the difficulty in creating illegitimate content (including illegitimate copies). For example, watermarks have been proposed to protect documents such as identification cards and passports [45].

- “Smart” Content [46]: An embedded watermark can be used in conjunction with devices to provide additional functionalities or services that benefit the user. For example, the watermark embedded in a music video may describe a link to the artist’s Internet site which allows a user to purchase the artist’s (other) works.
- Robust Data Hiding or Steganography [47]: The embedded watermark may be used as a covert channel to communicate messages from one user to another. For example, the sender embeds the watermark into a video, encoding the secret message in the watermark. The watermarked video is then provided to the recipient(s), possibly by using an insecure channel or by making the video publicly available. Because the watermarked video and original video are perceptually similar, the communication of the secret message is disguised by using the original video signal as an innocuous cover. The intended recipient(s) detects and decodes the watermark to obtain the secret information.

### **1.3.2 Synchronization and synchronization attacks**

A type of attack that has received considerable attention in watermarking is synchronization attacks. A synchronization attack confuses the watermark detector by moving, or re-positioning the embedded watermark. Unless the watermark detector can identify where the watermark resides in the attacked video, most watermark detectors will fail to detect the watermark. This is a major vulnerability in watermark detection. If the watermark cannot be detected, then the watermark does not confer

any benefit or protection in the application. It is often easier to confuse the watermark detector by a synchronization attack than to remove or destroy the embedded watermark, and in fact many synchronization attacks are easy, or even trivial, to perform.

The process of identifying the position of the watermark is known as watermark detector synchronization. Synchronization is generally a search to discover where the watermark has been re-positioned in a watermarked signal. While a blind, exhaustive search is not practical, a variety of strategies have been proposed to reduce the search. Some watermarking techniques confront synchronization issues by designing the watermark such that re-positioning is more difficult. Other techniques propose designing a pattern, known as a template, which allow the watermark detector to use an informed search to discover the position of the watermark. The ease by which synchronization attacks can render undetectable the watermarks produced by current watermarking techniques has inspired much effort to find techniques for efficient synchronization.

Efficient synchronization is the focus of this dissertation. Unlike most other works which merely propose a template or technique for synchronization, models of synchronization are developed that allow a deeper insight into why synchronization is relatively easy for some watermarks but considerably more difficult for others. In particular, when synchronization is viewed as an informed search, redundancy in the watermarked signal affects the difficulty of synchronization. The models also allow the development of watermarks that have demonstrable resilience against synchronization attacks.

#### **1.4 Overview of the Dissertation**

The primary objectives of this research are to obtain a deeper understanding of watermark detector synchronization and to design watermarks that are more robust against synchronization attacks.



Chapter 2 is a background of watermarking. The background includes an overview of the fundamentals of watermarking, digital image and video watermarking (which is the emphasis in this dissertation), and watermark synchronization.

Chapter 3 describes new models for temporal synchronization. These models encompass the behavior of many proposed video watermark embedders and detectors. Using the models, temporal synchronization is explored and the vulnerability of some current watermarking techniques against temporal synchronization attacks is demonstrated. The models also show that resilience against temporal synchronization attacks can be achieved by designing a watermark that possesses temporal redundancy. Experimental results corroborate the theoretical foundations.

In Chapter 4, spatial synchronization is explored. A method for efficient spatial synchronization is developed which is inspired from the temporal synchronization models. Spatial redundancy is used to construct an autocorrelation-based template for spatial synchronization. The results demonstrate that redundancy is helpful for spatial synchronization.

Chapter 5 describes some of the earlier work in semi-fragile watermarking for content authentication. The watermarking technique described in this chapter allows alterations to a watermarked image to be distinguished from more innocuous signal processing, such as lossy compression. Because digital signals can be manipulated with relative ease, tamper detection and authentication is useful, if not necessary, for the use of digital signals as evidence in law, journalism, intelligence, and other applications. This work is not related to watermark detector synchronization.

Conclusions are in Chapter 6, as well as directions for future work.

## 2. BACKGROUND

This chapter provides an overview of the fundamental concepts and current approaches in digital watermarking, particularly for video and still images. The focus of this research work is watermark synchronization, which will be introduced and identified as a significant issue (and vulnerability) in watermark detection.

The overview of watermarking begins with a description of watermark embedding, detection, and attacks. These three fundamental processes are described in Section 2.1. The desirable properties of watermarks and watermark evaluation are discussed in Section 2.2. While there is a slight emphasis towards the watermarking of images and video in these sections, the discussion is generally applicable for watermarking of multimedia signals, including audio. The structure and properties of digital images and video are described in Sections 2.3. Section 2.4 briefly describes digital video compression. Section 2.5 is a review of watermarking techniques specific to digital images and video, including techniques for watermarking compressed video. Synchronization is described in Section 2.6. Some topics useful in later discussions appear in Section 2.7.

### 2.1 Watermarking Overview

Digital watermarking [27–33] is the art of embedding information into an original signal in an imperceptible and secure manner, and the subsequent detection of the embedded information from the watermarked signal. The watermarked signal may be attacked, or altered, before it is made available to the watermark detector. These three processes of *watermark embedding*, *attack*, and *watermark detection* are fundamental to watermarking and are shown in Figure 2.1. Each of these fundamental processes will be elaborated below.

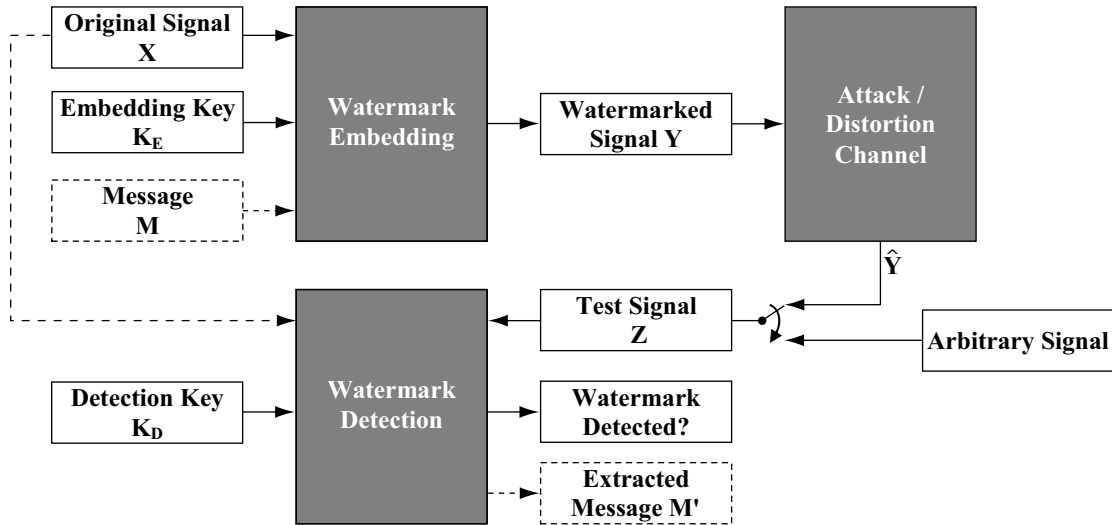


Fig. 2.1. Classical model of watermarking

### 2.1.1 Watermark embedding

Watermark embedding is the process of encoding information in the form of the watermark and then inserting the watermark into an original signal to produce the watermarked signal. The watermark embedder accepts as inputs the original signal  $X$ , embedding key  $K_E$ , and message  $M$ , and produces the watermarked signal  $Y$ . The watermarked signal contains the embedded watermark and is therefore not identical to the original signal. Nevertheless, the original signal and the watermarked signal should be perceptually similar under casual observation. Distinct signals may be perceptually similar because of limitations in human perception [32, 34, 48–50], which makes watermarking (and lossy compression [51–54]) possible for visual and auditory signals.

The original signal  $X$  is the signal into which the watermark will be embedded. Once the watermarked signal is produced, the original signal is safeguarded. The use of watermarking confers no protection or benefit to the original signal because the original signal is not watermarked. Unrestricted access to the original signal would defeat the purpose of watermarking.

The embedding key  $K_E$  is the secret that is necessary to generate and embed the watermark. A watermarking technique is generally capable of embedding many watermarks and is not limited to embedding a single, specific watermark. From the embedder's viewpoint,  $K_E$  is a parameter that identifies which watermark to embed. Each (choice of)  $K_E$  identifies a distinct watermark. From a broader perspective, knowledge of  $K_E$  confers the ability to embed the watermark. That is, embedding the watermark with knowledge of the corresponding  $K_E$  is easy. However, embedding the watermark without knowledge of  $K_E$  should be very difficult, even with complete knowledge of all the workings of the watermark embedder.

The set of all embedding keys is the embedding keyspace,  $\mathcal{K}_E$ , and  $K_E \in \mathcal{K}_E$ . The cardinality of the keyspace, or  $|\mathcal{K}_E|$ , is generally very large.

A detail that needs to be addressed when using watermarking is key management [16,55]. One aspect of key management is the assignment of keys to recipients, owners, users, and other entities in an application. For example, an application embedding watermarks for expressing ownership would assign a unique  $K_E$  to each owner. Key management, however, is not a concern in this dissertation. It is sufficient herein that each  $K_E$  corresponds to a distinct watermark.

Watermarking techniques may allow additional side information to be encoded into the watermark, known as the payload or message  $M$ . Upon successful watermark detection (see Section 2.1.3), the watermark detector is able to extract the payload and provide the extracted payload to the application. The same payload may be encoded in multiple watermarks (corresponding to distinct  $K_E$ 's). Some watermarking techniques do not support encoding a payload.

Watermark embedding is the two-step process shown in Figure 2.2. First, the watermark signal  $W$  is constructed by the watermark generator using  $K_E$  and  $M$ . While the specific methods by which  $W$  is constructed depends on the watermarking technique, many watermarking techniques produce  $W$  using a pseudo-random number generator (PRNG). Section 2.7.1 describes PRNGs in more detail. The

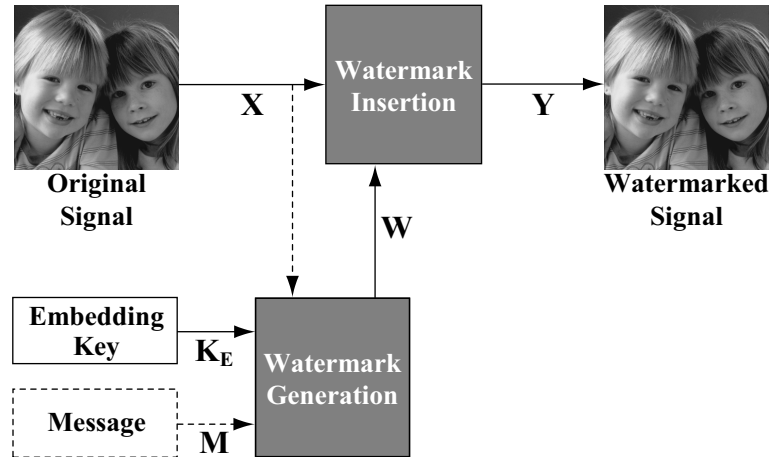


Fig. 2.2. Watermark embedding

embedding key  $K_E$  is typically the seed to the PRNG, which results in  $W$  as a  $K_E$ -dependent noise-like pattern.

Some watermarking techniques provide the original signal  $X$  to the watermark generator, which permits signal-adaptive embedding. Adapting the watermark to the characteristics of the original signal can significantly improve the performance of watermarking techniques. One way to exploit signal-adaptive embedding is to use a perceptual model [32, 34] to allow the watermarked signal to be more perceptually similar to the original signal. Signal-adaptive watermarks can also be more robust against attack [56, 57]. The concept of generating the watermark based on an examination of the original signal is known as informed embedding [28, 58, 59].

Once  $W$  has been constructed, it is inserted into the original signal in the step of watermark insertion. Watermark insertion introduces distortion into the watermarked signal, causing the watermarked signal to be distinct from the original signal. When the watermarked signal is shown or displayed, the effect of the distortion is similar to noise. Like watermark generation, the methods by which the watermark is inserted depends on the watermarking technique. Watermark generation and watermark insertion may be tightly coupled and are not necessarily independent processes.

Many watermarking techniques use the insertion methods of additive embedding, multiplicative embedding, or quantization embedding. In additive embedding,  $W$  is added into the corresponding sample values of  $X$  analogous to additive noise, or

$$Y = X + \gamma W, \quad (2.1)$$

where  $\gamma$  is a parameter which scales the amplitude of the embedded watermark. This parameter is commonly referred to as the embedding strength. Multiplicative embedding [60,61] uses

$$Y = (1 + \gamma W)X. \quad (2.2)$$

Quantization embedding is watermark insertion by quantizing samples of  $X$ . Quantization [62] is a non-reversible process which transforms a continuous value to one of a countable set of values. The difference between the quantized value and the original value is known as the quantization error. Different quantizers may be used for each sample or group of samples of  $X$ . Watermarking techniques using quantization embedding include Quantization Index Modulation (QIM) [63] and Scalar Costa Scheme<sup>1</sup> (SCS) [56,64,65] watermarks. When quantization error is viewed as additive noise, then for quantization embedding  $W$  is the residual  $W = Y - X$  that has been added  $X$  to produce  $Y$ .

Before moving the discussion to attacks, an example is useful to illustrate the basic concepts of watermark embedding. This example uses the still-image watermarking technique described in [61]. The watermark is a pseudo-random sequence multiplicatively inserted in the Discrete Cosine Transform (DCT) [66,67] coefficients of the *Fruit* image shown in Figure 2.3. The embedding strength is varied. Figure 2.4 shows the watermarked images and the corresponding difference images for  $\gamma = 0.01, 0.1, 0.2$ , and  $0.5$ . The difference images show the difference between the corresponding pixels of the watermarked image and the original image. A neutral gray region in the difference image indicate that the corresponding pixel values of the

---

<sup>1</sup>SCS watermarking extends the earlier work in QIM, hence QIM with dither modulation is a special case of SCS.



Fig. 2.3. Original image *Fruit*

watermarked and original images are similar, while bright and dark regions indicate dissimilar pixel values. The contrast of the difference images have been increased to make the differences more visible.

The watermarked images show that as the strength is increased, the watermark becomes more noticeable. For example, the distortion in the watermarked image is generally noticeable for  $\gamma = 0.5$ . However, if the strength is sufficiently small then there is little or no perceptual difference between the watermarked image and the original image. The inserted watermark is more noticeable in relatively smooth regions of the image, such as the background region near the very top of the image, and less noticeable in “busy” regions of the image such as the region near the grapes and strawberries.

### 2.1.2 Watermark attack

The watermarked signal may be attacked [68–70] before examination by the watermark detector. An attack is any process which may remove the embedded watermark, increase the difficulty in detecting the watermark, or subvert the security of the watermark. While the word “attack” has an adversarial connotation, attacks do not necessarily have malicious intent. For example, the watermarked signal may be subjected to innocuous processing in an application. Conversely, attacks may arise from deliberate attempts to remove an embedded watermark or defeat the security

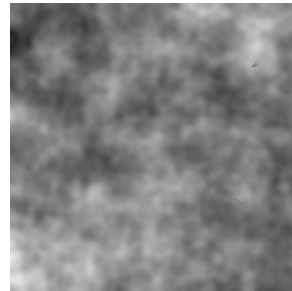
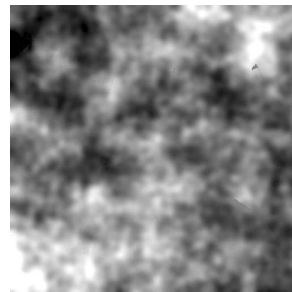
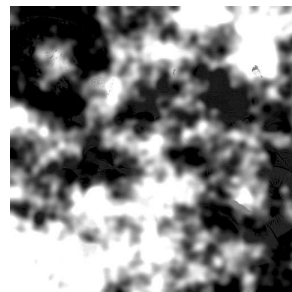
(a) Watermarked  $\gamma = 0.01$ (b) Difference  $\gamma = 0.01$ (c) Watermarked  $\gamma = 0.1$ (d) Difference  $\gamma = 0.1$ (e) Watermarked  $\gamma = 0.2$ (f) Difference  $\gamma = 0.2$ (g) Watermarked  $\gamma = 0.5$ (h) Difference  $\gamma = 0.5$ 

Fig. 2.4. Watermarked *Fruit* and difference images using technique of [61]. Distortion introduced by watermark insertion is more visible as the strength  $\gamma$  is increased.



of the watermarking system. The watermarked signal may also be attacked multiple times.

Removal attacks are attacks which damage or destroy an embedded watermark. Removal attacks include filtering, lossy compression, addition of noise, noise removal, quantization [71], transcoding [72, 73], amplitude scaling [74], and digital-to-analog and analog-to-digital conversion. Some removal attacks (such as lossy compression) are general processing which do not specifically target the embedded watermark but may incidentally destroy or damage it. Other removal attacks are designed to eliminate the embedded watermark while preserving the “quality” or usefulness of the attacked signal. Some removal attacks directed specifically against watermarks are described in [70, 75–81].

Another method for attack is to increase the difficulty of detecting the watermark. These detection-disabling attacks do not remove the watermark, but obfuscate the watermark from the watermark detector.<sup>2</sup> Because the watermark is not removed, the watermark could be readily detected by the watermark detector if the detector can determine (or if it is provided with information) how the watermark has been obfuscated. One type of detection-disabling attack that has received considerable attention is synchronization attacks. Watermark detector synchronization and synchronization attacks are the focus of this dissertation and will be discussed in Section 2.6. Another detection-disabling attack is the Mosaic Attack [70]. In the Mosaic Attack, a large watermarked signal is partitioned into smaller signals, each of which is sufficiently small such that the watermark detector will fail to detect the watermark when examining the small signal alone. The Mosaic Attack is effective against watermark detectors that do not possess the ability or resources to splice the smaller signals and detect the watermark on the unified signal. An example of the Mosaic Attack is shown in Figure 2.5.

---

<sup>2</sup>While a detection-disabling attack is not directed towards watermark removal, the manipulations to the watermarked signal that are performed as part of the detection-disabling attack may, as a secondary effect, damage or destroy an embedded watermark.

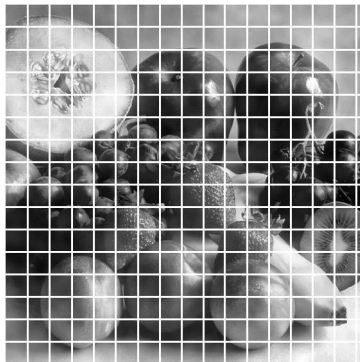


Fig. 2.5. Mosaic Attack on the watermarked *Fruit* image. Each block represents an individual image. The gaps between the images are not present when the images are displayed. Watermark detection fails when each of the small images is examined individually.

Collusion attacks exploit correlation to remove embedded watermarks. Correlation may be present in multiple signals or within a single signal [82, 83]. When multiple watermarked signals are produced from a common original signal, the watermarked signals are correlated because they all contain the original signal. The only difference amongst the watermarked signals are their embedded watermarks, which are usually uncorrelated with respect to each other. Averaging a sufficient number of watermarked signals provides a good estimate of the (unwatermarked) original signal. This is a significant issue for content tracking, where a group of users may conspire to create an unwatermarked signal which does not identify any of the conspirators. There have been efforts in designing collusion-secure watermarks [82–87] which can expose one or more conspirators when the total number of conspiring users is sufficiently few. But despite these efforts, collusion remains a significant problem for content tracking. Collusion attacks may also exploit correlation within a signal. This latter type of collusion attack is an issue for video watermarking and will be described in Section 2.5.2.

Not all attacks are concerned with removing or obfuscating embedded watermarks. Some attacks, such as security and protocol attacks [69], are designed to

subvert watermark security. Two attacks are briefly mentioned as examples: the copy attack and the ambiguity attack.

The copy attack [88] is a method of forgery by estimating and transferring the watermark from a watermarked signal to another (arbitrary) target signal. The copy attack can have significant repercussions in watermarking applications. For example, when watermarks are used for content tracking, an innocent user may be framed when the copy attack is used to transfer the innocent user's watermark from a legitimate signal to a pirated signal. When watermarks are used to express ownership, the copy attack may be used to transfer an owner's watermark from an innocuous work into disturbing or illegal works, such as child pornography. Countermeasures against the copy attack include signal-adaptive watermarking, such as [89], and also [90–92].

The ambiguity attack is an attack against copyright or ownership watermarks [93–95]. An ownership watermark is embedded so that, in the event of a dispute, the owner can demonstrate ownership by detecting his or her own watermark in the disputed content. However, an attacker may also produce another signal, a “false original,” in such a way that the true owner cannot be determined by the watermark. One way to combat the ambiguity attack is to use non-inverible watermarking [93] to make difficult the creation of a false original. See also [94, 96], and embedding secure timestamps [16].

The context of the application is an important consideration when discussing attacks. While many attacks against watermarks have been described, not all of the attacks are necessarily significant for a particular application. For example, the application may anticipate certain kinds of attacks and require an embedded watermark to be robust to these attacks. Robustness against other attacks may have lesser or no importance, particularly if such attacks are not likely to occur in the application. This will be mentioned again when security is discussed in Section 2.2.4.

### 2.1.3 Watermark detection

The last fundamental process to be introduced is watermark detection. The objective of the watermark detector is to determine whether or not an input signal  $Z$  is watermarked. The watermark detector is also provided with a detection key  $K_D$ , and, in some techniques, the original signal  $X$ . If watermark detection is successful and the technique supports a payload, the watermark detector also extracts the payload  $M'$ . If  $Z$  had been attacked, then  $M'$  is not necessarily identical to  $M$ .

The detection key  $K_D$  specifies the watermark to be detected. An embedded watermark should only be detectable with knowledge of  $K_D$ . If the key to detect the watermark is identical to the key used to embed the watermark ( $K_D = K_E$ ), then the watermarking technique is known as a symmetric or private key watermarking technique. Otherwise,  $K_D \neq K_E$  and the technique is known as an asymmetric or public key watermarking technique [41, 97, 98]. Asymmetric watermarks have the advantage that the knowledge to detect a watermark does not confer the ability to embed the watermark (because a different key is used.) The vast majority of proposed watermarking techniques are symmetric.

The detection keyspace  $\mathcal{K}_D$  is the set of all detection keys, and  $K_D \in \mathcal{K}_D$ . For symmetric techniques, the embedding and detection keyspaces are identical and the keyspace of the technique can be written as  $\mathcal{K} = \mathcal{K}_E = \mathcal{K}_D$ . The embedding and detection keyspaces may be identical or distinct for asymmetric techniques.

Some watermark detection techniques require access to the original signal for watermark detection. These techniques are known as non-blind (detection) techniques. A non-blind technique requires a secure method for providing the original image to the watermark detector, which is a significant limitation in many proposed applications. Watermarking techniques that do not require the original signal for watermark detection are known as blind (detection) techniques.

In general, the input test signal  $Z$  is arbitrary and can be any signal. However, four cases are considered when describing the desired output of the watermark detector:

1. When  $Z$  is the original signal  $X$ .
2. When  $Z$  is the watermarked signal  $Y$ .
3. When  $Z$  is a watermarked signal that has been attacked  $\hat{Y}$ .
4. When  $Z$  is any other signal.

Because the original signal is not watermarked, the watermark detection should fail when the original signal  $X$  is examined by the detector. Conversely, watermark detection should always succeed if the watermarked signal  $Y$  is examined and the proper detection key is provided to the detector. If the watermarked signal has been attacked, the watermark detector should successfully detect the watermark unless: (1) the attack sufficiently damages or degrades the watermarked signal so that it is no longer usable in the application, in which case the output of the detector is “don’t care,” or (2) the watermarking technique is fragile. The focus of the discussion is on robust watermarks, which are watermarks designed to be detectable even after the watermarked signal has been attacked. Fragile watermarking will not be discussed until Chapter 5. The watermark should not be detected if any other signal is examined by the detector because these signals are not watermarked. Table 2.1 summarizes the desired output of an ideal robust watermark detector.

When the watermark detector examines  $Z$  and produces a result (watermark present, watermark not present), its correctness is one of three cases: correct, false positive, or miss. Assuming that the corresponding  $K_D$  is provided to the detector, the watermark detector is correct when it reports  $Z$  as watermarked and  $Z$  (actually) is watermarked. The watermark detector is also correct if it reports that  $Z$  is not watermarked and  $Z$  is (actually) not watermarked. A false positive occurs when the watermark detector reports an unwatermarked  $Z$  as watermarked. A miss occurs

Table 2.1  
Desired output of ideal robust watermark detector

Test Signal $Z$	Detection Key	Detector Output
Watermarked, not attacked ( $Y$ )	Correct $K_D$	Watermark Present
Watermarked, not attacked ( $Y$ )	Incorrect $K_D$	Watermark Not Present
Watermarked, attacked ( $\hat{Y}$ )	Correct $K_D$	Watermark Present*
Watermarked, attacked ( $\hat{Y}$ )	Incorrect $K_D$	Watermark Not Present
Original signal ( $X$ )	Any	Watermark Not Present
Other (none of the above)	Any	Watermark Not Present

\*If the attack sufficiently damages or degrades the watermarked signal such that it is no longer usable in the application, the detector output is “Don’t Care”.

when the watermark detector reports watermarked  $Z$  as unwatermarked. Obviously, false positives and misses are detection errors. When any other  $K_D$  is provided to the watermark detector, a false positive occurs when the detector reports  $Z$  as watermarked. A miss is not possible because the watermark detector should never detect an embedded watermark without the proper  $K_D$ .

The watermark detector is able to detect an embedded watermark and extract the payload because it has access to side information. Specifically, (1) the watermark detector is aware of the specific methods by which the watermark is created and embedded, and (2) the watermark detector is provided with the detection key. This side information generally provides the detector with complete information how the watermark has been embedded.<sup>3</sup> Using the side information, the detector examines the test signal  $Z$  and determines the likelihood that  $Z$  is watermarked by using fundamental signal detection and statistical signal processing principles [28, 99–103]. While not all watermark detectors use correlation (for example, [104, 105]), correla-

---

<sup>3</sup>Thus, the signal  $W$  is often available to the watermark detector, even though  $W$  is not an explicit input to the detector.

tion or matched filtering followed by a decision threshold is very common. Because the watermark detector makes use of knowledge how the watermark is embedded, the watermark detector is usually specific to the watermarking technique.

The two main challenges for robust watermark detection are to detect the watermark in an attacked signal and to prevent interference from causing detection error. Robustness against attack is often addressed by modeling. Developing and using models allow countermeasures to be devised against the attacks in which the model is applicable. Unfortunately, models are generally applicable to a relatively small set of attacks against watermarks. For example, a model suitable for some removal attacks may not consider detection-disabling or synchronization attacks. Designing a watermark that is robust against “everything” remains elusive because currently there is no useful model that describes all attacks against watermarks.

Watermark detection is affected by interference, which may cause detection error even in the absence of attack. Two sources of interference are host-signal interference and watermark interference. Host-signal interference is the phenomenon where the original signal acts like a source of noise with respect to blind watermark detection. Because the watermark should be perceptually invisible, the power of the signal of interest to the detector (the watermark) is often orders of magnitude less than the power of the interference source (the original signal). If the original signal is available at the detector, or non-blind detection, then the host-signal interference is known and can be subtracted or removed by the detector. Table 2.2 summarizes this discussion. Host-signal interference can be reduced in blind detection by informed embedding [28, 56, 64, 106–108]. Filtering or whitening may also reduce host-signal interference [38, 109].

Watermark interference occurs when two or more watermarks produced by distinct  $K_E$ 's appear indistinguishable to the watermark detector. In other words, the one-to-one correspondence between  $K_E$  and  $K_D$  is lost. This interference may occur coincidentally or as a result of poor design of the watermarking technique. The result

Table 2.2

Observation of the watermarked signal  $\hat{Y}$  by users and watermark detectors. The user is interested in the original signal. Watermark detectors are interested in the watermark. A non-blind detector has access to the original signal and can remove host-signal interference.

When $\hat{Y}$ is examined by	Signal of Interest	Noise	Effect of Noise
User	$X$	$W + \text{Attacks}$	Perceptual distortion; may be distracting
Blind Detector	$W$	$X + \text{Attacks}$	Attacks or host signal interference may cause miss
Non-Blind Detector	$W$	Attacks	Attacks may cause miss

is false positives, because a watermark produced by any of the affected embedding keys will be detected by using any of the corresponding detection keys.

This section concludes with an example of a simple additive watermarking technique with a blind correlation-based detector. This example is similar to the watermarking technique implemented in the experimental work described in Chapter 3. Suppose  $X = \langle x[0], \dots, x[N-1] \rangle$  is an original signal of length  $N$  samples.<sup>4</sup> The watermark is generated as a  $K_E$ -dependent, zero mean,  $N$ -dimensional signal  $W = \langle w[0], \dots, w[N-1] \rangle$  and additive embedding is used to produce the watermarked signal  $Y = X + \gamma W$ . The watermark detector is a blind correlation-based detector which obtains the (linear) correlation:

$$\rho = \frac{1}{N}(Z \cdot W) \quad (2.3)$$

The dot ( $\cdot$ ) signifies dot-product, or the sum of the products of corresponding samples. If  $\rho$  is sufficiently large, then the watermark detector concludes that the water-

<sup>4</sup>In this text, the notation  $\langle \cdot \rangle$  is used to denote an ordered sequence, such as the samples of a signal or the elements of a vector, and  $\{ \cdot \}$  is used to denote set membership.  $\mathbf{X} = \langle x_1, x_2, x_3 \rangle$  is the ordered sequence  $x_1$  followed by  $x_2$ , followed by  $x_3$ .  $\mathcal{X} = \{x_1, x_2, x_3\}$  is a set with members  $x_1$ ,  $x_2$ , and  $x_3$ .



mark is present. Otherwise, the watermark detector concludes that the watermark is not present. In signal detection theory, this is expressed as a comparison with some detection threshold  $T$ :

$$\text{Detector Output} = \begin{cases} \rho \geq T : & \text{Watermark Present} \\ \rho < T : & \text{Watermark Not Present} \end{cases} \quad (2.4)$$

A common assumption in watermarking is that an arbitrary signal is not correlated with the watermark. This assumption is based on the fact that watermarks typically possess a noise-like structure that does not resemble any useful signal. Thus, given an arbitrary unwatermarked signal  $Z$ , the expected correlation value is

$$\text{E}[\rho|Z \neq Y] = \frac{1}{N} \text{E}[Z \cdot W] = 0 \quad (2.5)$$

Now suppose that the input is watermarked, or  $Z = Y$ . Then, substituting in (2.3),

$$\begin{aligned} \rho &= \frac{1}{N} (Y \cdot W) \\ &= \frac{1}{N} ([X + \gamma W] \cdot W) \\ &= \frac{1}{N} (X \cdot W) + \frac{\gamma}{N} (W \cdot W) \end{aligned} \quad (2.6)$$

and taking expectation,

$$\text{E}[\rho|Z = Y] = \frac{1}{N} \text{E}[X \cdot W] + \frac{\gamma}{N} (W \cdot W). \quad (2.7)$$

Making a similar assumption that  $W$  is not correlated with arbitrary (original) signal  $X$ , then

$$\text{E}[X \cdot W] = 0 \quad (2.8)$$

and the expected correlation value becomes

$$\text{E}[\rho|Z = Y] = 0 + \frac{\gamma}{N} (W \cdot W) = \frac{\gamma}{N} (W \cdot W) \quad (2.9)$$

Equation (2.9) is the expected correlation value when the input is watermarked. The threshold value of (2.4) is chosen to be less than (2.9), such as  $T = \frac{\gamma}{2N} (W \cdot W)$ , although a precise value for  $T$  is not important in this example. In practice,  $T$

would be chosen based on the desired false-positive or miss rate, if such information is available or can be estimated [99–101].

Host-signal interference is expressed by the first term of (2.6). This equation shows that the original signal  $X$  appears in  $\rho$ , which allows  $X$  to interfere and cause detection error. The assumption of (2.8) is that for arbitrary  $X$ , the expectation is that  $X$  is not correlated with  $W$ . However, there will generally be some correlation (but usually weak correlation) between a *fixed*  $X$  and  $W$ , thus  $X \cdot W$  will be non-zero. This may cause  $\rho$  to fall beneath threshold  $T$  and the detector to miss, particularly if some part of  $X$  is (coincidentally) anti-correlated with  $W$ . When the embedding power of  $W$  is low, the second term of (2.6) is decreased, which increases the influence of the first term (the interference term). The expectation of (2.5) also only applies for arbitrary unwatermarked  $Z$ , and there will generally be some correlation between a fixed  $Z$  and  $W$ . If the correlation is sufficiently large, than  $\rho$  will be greater than the threshold, causing a false positive for  $Z$ . These detection errors may occur even without attacks.

## 2.2 Watermarking Objectives and Evaluation

There is general agreement that the performance criteria [28–33, 110, 111] for watermarking should include at least perceptual transparency, robustness, capacity, and security. Perceptual transparency is the degree of invisibility of the embedded watermark when the watermarked signal is displayed. Robustness is the resilience of the embedded watermark against removal by signal processing. Capacity is the amount of information that can be encoded or expressed by the watermark, and security is the ability of the watermark to resist hostile attacks.

### 2.2.1 Perceptual transparency

Perceptual transparency is the degree of invisibility of the watermark when the watermarked signal is displayed. Watermark embedding requires inserting distortion

into a signal, and as a result watermarking is not possible if no amount of distortion is tolerable. While the precise limits of watermark visibility is determined by an application, preserving the “quality” of the watermarked signal and minimizing the perceived distortion caused by watermark embedding are generally desirable.

One way to quantify distortion is the mean-square error. The mean-square error between any signals  $S_1$  and  $S_2$  is defined as

$$\text{MSE}(S_1, S_2) = \frac{1}{N} \sum_{\forall \mathbf{v}} (S_1[\mathbf{v}] - S_2[\mathbf{v}])^2 \quad (2.10)$$

where  $\mathbf{v}$  is a coordinate vector ( $t$  for one-dimensional signals,  $(x, y)$  for two-dimensional signals, and  $(x, y, t)$  for three-dimensional signals) and  $N$  is the total number of samples in each signal. When  $S_1$  and  $S_2$  are identical, then  $\text{MSE}(S_1, S_2) = 0$ . A related distortion measure is the peak signal-to-noise ratio, or PSNR, in decibels (dB)

$$\text{PSNR}(S_1, S_2) = 10 \log_{10} \left[ \frac{(S_{max})^2}{\text{MSE}(S_1, S_2)} \right] \quad (2.11)$$

where  $S_{max}$  is the maximum value for any sample of  $S_1$  or  $S_2$ . The higher the  $\text{PSNR}(S_1, S_2)$ , the less distortion between  $S_1$  and  $S_2$ . If the signals are identical, then  $\text{PSNR}(S_1, S_2) = \infty$ . The MSE and PSNR provide a way of quantifying the true distortion between two signals. However, true distortion is not as significant in watermarking as perceptual distortion, or how the distortion is perceived when an observer views the distorted signal. Neither the MSE nor the PSNR consider how distortion is perceived.

When examining the perceptibility of distortion, the spatial and temporal distribution of the distortion is as significant as the total power of the distortion. For example, distortion that alternates rapidly in successive frames of video may appear to flicker or shimmer to an observer. This can be extremely distracting, even when the total amount of distortion is small. On the other hand, relatively large amounts of distortion may be unnoticed in textured or busy areas of an image, such as the flowers in *Flowergarden* (Figure 2.6). In general, many factors affect human perception of distortion [34, 50, 112, 113], including the (original) signal characteristics, distortion characteristics, and the environmental or viewing conditions.



(a) Original



(b) Noisy

Fig. 2.6. Visibility of distortion in *Flowergarden*. Additive white noise has been inserted into the image. The noise is independent of the image and uniformly distributed with zero mean. The PSNR of the noisy image is 27.67 dB. The distortion is not very noticeable at the bottom (textured flower region) of the noisy image but obvious at the top (smooth sky region).

There have been efforts to develop perceptual distortion measures that are applicable for assessing image and video quality, including [112,114–121]. These distortion measures vary in complexity and accuracy (or agreement) with subjective evaluation. Unfortunately, perception of visual distortion is an open problem and there is currently no distortion measure that agrees with subjective assessment for all types of images, types of distortion, and viewing conditions. The PSNR is often used as an objective measure of distortion even though the PSNR does not consider perception and does not fully agree with subjective assessment of visual quality [119]. The widespread use of the PSNR in the watermarking (and lossy compression) literature does allow some “loose” basis for comparison. Until there is agreement on a perceptual distortion measure to assess visual quality, watermarking techniques typically select (perhaps arbitrarily) one or more known distortion measures, such as the MSE, PSNR or some perceptual distortion measure, to demonstrate perceptual transparency.

### **2.2.2 Robustness**

Robustness is the resilience of the embedded watermark against removal by signal processing. A watermark is robust against an attack if the attack does not appreciably remove the watermark or hinder its detection by the watermark detector. Many applications do not require the watermark to be robust against any conceivable attack, but only against attacks that are likely or expected to occur before the watermarked signal is examined by the watermark detector. Robustness is also not necessary against attacks that sufficiently damage the watermarked signal to render the attacked signal useless.

A variety of methods have been used to experimentally demonstrate robustness against attacks. Some papers describing techniques using correlation-based detectors show the detector correlation values with and without attack. The watermark is robust if the correlation values do not change appreciably after attack. For techniques

that support a payload, robustness can be measured as the bit error rate (BER) of the extracted payload. A robust technique will have a very low payload BER after attack. In other techniques, the detector examines a large input signal and detects the watermark on a unit basis, such as a video watermark detector that performs a separate detection on each video frame. In these techniques, robustness can be measured by the average number of successful detections, or the average number of successful detections per unit time. The detection rate of a robust technique does not appreciably decrease after attack. Because there are many methods to operationally measure robustness, comparing the robustness of different watermarking techniques can be awkward.

### 2.2.3 Capacity

Capacity is the amount of information that can be represented or encoded by an embedded watermark. For watermarking techniques that do not support a payload, the capacity is a single bit (watermark is present or watermark is absent). Watermarking techniques which support a payload [122] typically express the capacity in terms of bits of payload per sample of the watermarked signal. A technique with higher capacity allows more information to be embedded in a signal of fixed size, and is generally desirable. A signal-dependent watermarking technique may embed more information when the characteristics of the original signal are favorable, and less information otherwise.

Capacity and robustness may be viewed from the perspective of a communications framework [28, 63, 123]. In this view, the watermark is analogous to a channel which communicates the payload from a sender (the watermark embedder) to a receiver (the detector). Without any attack, the capacity of the channel is a property of the watermark technique and, in the case of blind techniques that are affected by host-signal interference, the original signal. The effect of attacks is to reduce the capacity of the watermark, until capacity reaches zero and no information may be

reliably communicated. A robust watermark is one whose capacity is not significantly reduced as a result of attack.

#### 2.2.4 Security

Watermark security [28, 124] is the ability of the watermark to resist hostile attacks. Attacks can render the watermark useless or subvert how the watermark is used in an application, thereby compromising watermark security. In general, watermark security is compromised if the embedded watermark can no longer function as intended by an application. The application determines the (possible) threats against the watermark, the resources (including technical knowledge) possessed by adversaries, and the cost or consequences of security breaches. Some applications do not require much security, while other applications anticipate many threats against an embedded watermark and require a high level of security.

While the security requirements of specific applications lie beyond the scope of this discussion, applications generally consider one or more of the following threats: unauthorized embedding, unauthorized detection, and unauthorized removal [28]. Unauthorized embedding, or forgery, is an attack which embeds a false watermark into an arbitrary signal or allows the payload of an existing watermark to be altered. The copy attack is an example of unauthorized embedding. Unauthorized detection is the detection of the watermark by users which should not be able to detect the watermark. An example of unauthorized detection is estimation of the watermark by processing a watermarked signal. Unauthorized removal refer to attacks which remove an embedded watermark or prevent the watermark from being detected. Examples of unauthorized removal include removal, detection-disabling, synchronization, and collusion attacks.

The threats of unauthorized embedding, detection, and removal imply that estimating or deducing the structure of an embedded watermark (by examination or processing a watermarked signal) should be very difficult. If an attacker can success-

fully deduce the structure of a watermark, then removing the watermark is trivial (for example, [78]). The attacker may also be able to transfer the watermark to another signal by the copy attack.

Security is not the same concept as robustness. While many watermarks must be robust to be secure, this is not always the case. Some watermarks can be secure without being robust. These watermarks will be described in Chapter 5. Conversely, robustness alone may not be sufficient for security. For example, a robust watermark may be vulnerable to the copy or ambiguity attacks, and hence insecure for some applications. Some applications require watermarks with a high degree of robustness but relatively little security, for example [46]. The security requirements of [46] are relatively low because users have little reason to attack the watermark. However, this application requires a watermarking technique with robustness to many signal processing attacks. These attacks include noise insertion, geometric distortion, changes in image brightness, and many other effects that arise from the printing and detection processes which are used in [46].

### 2.2.5 Other criteria

In some proposed applications, the watermark embedder resides in an environment where available (computational) resources are scarce, or where real-time watermark embedding is needed. One example is watermarking of broadcast video of a live performance or sporting event. In these applications, watermark embedding is constrained by the available resources and the cost of watermark embedding is a significant issue. Other applications, such as content tracking, require many different watermarked signals to be produced. Summing the cost to produce each watermarked signal can result in a significant total cost. This cost is not necessarily in computation. For example, content tracking of printed images requires a print process that can produce a large number of unique (watermarked) images. Such



a print process is generally more expensive than the process of printing the same number of identical copies of a single image.

Similarly, the cost of watermark detection is incurred each time the watermark detector examines an input signal. Detection cost is a concern for applications that require frequent watermark detection, such as for each time a signal is accessed, copied, or displayed [14]. The detection cost is also a concern if the watermark detector is constrained by limited resources or when a quick response time is required [46]. Broadcast monitoring is an example of an application which performs many watermark detections. An example of an application in which detection cost may not be critical is content tracking. In content tracking, watermark detection is performed only when needed and any reasonable cost is generally acceptable.

Detection accuracy refers to the false positive and miss rates of the watermark detector. It was mentioned in Section 2.1.3 that detection error can occur even without attack, such as errors caused by host-signal and watermark interference. Thus, detection accuracy can be examined under attack-free conditions as well as when the watermarked signal is attacked. It is generally not feasible to evaluate detection accuracy by empirically testing over the set of all original signals and the set of all watermarks because the cardinalities of these sets are generally large. However, evaluation is possible by modeling [28].

### **2.2.6 Performance evaluation, benchmarking, and tradeoffs**

Benchmarking and performance evaluation [70, 125–130] are open issues in watermarking. Common objectives of watermark benchmarking are to (fairly) compare the performance of watermarking techniques and to determine the suitability of watermarking techniques to the requirements of an application. For a variety of reasons, these objectives are difficult to achieve. First, as the earlier discussion has mentioned, there is no clear agreement on how some of the performance criterion should be objectively measured and compared. Second, even if objective measures are obtained

for each criterion, there is no agreement how the measures should be combined to a holistic view that can be compared to the needs of a specific application.

A third reason why watermark benchmarking is difficult is that the design of watermarking techniques represent tradeoffs in the performance criteria. One can trade, or sacrifice, performance in one or more criterion to improve the performance in another criterion. For example, decreasing the embedding strength (or equivalently, the embedding power) of the watermark allows perceptual transparency to be improved at the cost of decreased robustness. This tradeoff requires neither changing the specific methods used by the watermarking technique, nor changing any parameter other than the embedding strength. Some tradeoffs require modifying the watermarking technique but in a relatively simple manner, such as decreasing watermark capacity to increase robustness by encoding the payload using error-correcting codes [131] prior to embedding. These tradeoffs imply that a performance comparison should not consider only a specific “operating point” amongst the performance criteria, but rather the range of possible “operating points” that a technique can achieve. An analogy can be made with lossy compression: In lossy compression, a signal can be encoded at various data rates, and at each rate with a corresponding distortion [132, 133]. Comparing either the rate or the distortion alone is meaningless. However, the (achievable) rate-distortion performance is a characteristic of each lossy compression technique which allows comparison. This overall methodology for comparing techniques, analogous to rate-distortion analysis in lossy compression, has not been developed for watermarking.

Despite these issues, applications generally favor some (subset) of the performance criteria. The remaining criteria are of secondary importance, even if the relative importance may not be precisely quantified by some measure at this time. For example, an application may require robustness against certain attacks, regardless of any reasonable cost in security or capacity. Thus, certain tradeoffs are interesting for these applications, with complementary interest in techniques to achieve these tradeoffs with minimal cost.

### 2.3 Digital Images and Video

An image  $I(x, y)$  is a signal whose values represent the intensity of light emitted to an observer at each spatial coordinate  $(x, y)$ . An analog image is a continuous-space signal taking on a real value at any coordinate. A digital image [50, 113, 134–136] has values only at discrete coordinates, known as samples or pixels. Each pixel value is a member of a countable set. From an analog image, a digital image can be obtained in two steps: First, the analog image is sampled to form a discrete-space signal. Sampling [137, 138] does not cause any information to be lost if the analog image is band-limited and the sampling rate is sufficient (exceeding the Nyquist rate). After sampling, each sample value is quantized to one of a countable set of values and the digital representation is obtained. Many images use 8 bits to represent each pixel value, which allows for up to 256 discrete intensity levels.

An elementary view of digital video [50, 136, 139] is an ordered sequence of digital images that are displayed in succession, as well as the corresponding audio and synchronization signals. Each image of the video is known as a frame. The number of displayed frames per unit time is the frame rate. Some videos represent each frame as two separate fields [139] that are displayed in an interlaced or interleaved fashion. Fields more closely resemble the interlaced scanning used by analog television. The synchronization signal is used to maintain consistency during the presentation of the video, ensuring that the visual and audio signals are displayed together at the correct time. (This synchronization signal is not related to watermark detector synchronization.) While the focus of this discussion is the visual portion of the video signal, the audible portion is recognized as an integral part of the user experience for many videos.

The perception of color [48, 140–142] occurs when the observer's visual system and brain interpret light incident upon the retina with wavelengths of approximately 400 nm to 700 nm. The human retina has four types of receptors, with three types involved in color vision. Each type of color receptor has a different frequency response

to the power spectra of the incident light. (The frequency responses of the receptors generally overlap, such that a specific wavelength of light excites all three types of receptors but to different degrees.) From the receptors, a complex neural process occurs in the retina, optic nerve, and brain that eventually leads to the percept of color.

Previous studies have demonstrated that human color vision is trichromatic [142]. In the trichromatic model, a color can be obtained by superimposing, or mixing, three monochromatic lights. The monochromatic lights are known as primaries. The relative intensities of the primaries are adjusted to obtain or match a desired color.<sup>5</sup> As a result of the trichromatic model, a color image can be represented as the superposition of three monochromatic images. In particular, the entire spectrum between 400 nm and 700 nm does not need to be represented or encoded in a color image.

There are many different color systems, or colorspace, that can be used to represent a color image [136, 139, 142]. A colorspace is analogous to the coordinate system used to specify a color. Displays often use a colorspace based on the intensity of “red,” “green,” and “blue” light added to produce a color, known as an RGB colorspace [143, 144]. Colorspaces used in image and video processing typically use a luminance component and two chrominance components. The luminance is defined as radiant power weighted by the spectral sensitivity function that is characteristic of the human visual system [136], although “brightness” is often used as a synonym for luminance in non-technical discussions. The chrominance components specify the hue. A colorspace that is often used in digital video processing and compression is the  $YC_B C_R$  colorspace [145], which uses luminance  $Y$  and two chrominance components  $C_B$  and  $C_R$ . Linear and non-linear transformations allow a color expressed in one colorspace to be expressed in another colorspace.

---

<sup>5</sup>The range of colors that can be produced by the superposition of light is limited and depends on the primaries. This limitation arises because light can be “added” (by superposition) but not “subtracted.” The range of colors that can be produced using a fixed set of primaries is known as the gamut.

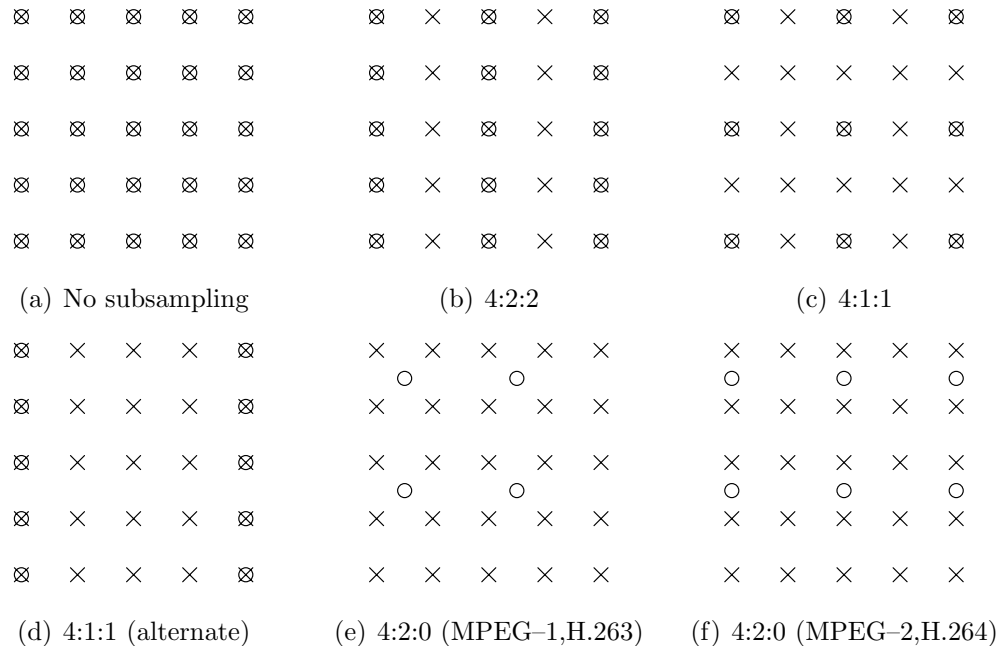


Fig. 2.7. Chrominance subsampling showing positions of chrominance pixels relative to luminance pixels ( $\times$ =Luminance pixel,  $\circ$ =Chrominance pixel)

In many digital color images, the luminance and chrominance images are sampled at the same rate, such that the luminance and chrominance pixels at  $(x, y)$  are co-located. However, the chrominance may be sampled at a lower rate and at distinct positions from the luminance in some images. The human visual system has less spatial acuity for color information than for brightness [136], and subsampling the chrominance allows a digital color image to be represented using less bandwidth. Chrominance subsampling is very common in digital video, and there are many ways to subsample. Figure 2.7 shows some examples, indicating the positions of the chrominance pixels ( $\circ$ ) relative to the luminance pixels ( $\times$ ).

The size of an image or video frame can refer to either the displayed size or the pixel size. The display size is the physical dimensions of the image when it is displayed. Display size is independent of pixel representation and is well-defined for both analog and digital images. The ratio of the display width to the display height

Table 2.3  
Common sizes of digital video frames

<b>Name</b>	<b>Luminance Pixels per Line</b>	<b>Luminance Number of Lines</b>
Sub-QCIF	128	96
QCIF	176	144
CIF*	352	288
4CIF	704	576
16CIF	1408	1152
CCIR601 [145]	720	480
SMPTE 274M [146]	1920	1080
SMPTE 296M [147]	1280	720

\*Common Intermediate Format

is known as the aspect ratio [136]. Conventional analog television uses a 4:3 aspect ratio while high definition television (HDTV) uses an aspect ratio of 16:9. Motion pictures typically use aspect ratios of 1.85:1 or 2.35:1. The pixel size is the number of pixels in the horizontal and vertical dimensions of a digital image. (Analog images do not have pixels and the pixel size undefined.) The ratio of the pixel width to the pixel height is not necessarily identical to the aspect ratio; pixels are not necessarily “square” when the video is displayed. In this dissertation, the image size shall refer to the pixel size unless stated otherwise. Some standard (pixel) sizes for digital video frames are shown in Table 2.3.

A challenge for processing or storing digital video is the rate of the video data. An uncompressed 4:2:2  $YC_B C_R$  CCIR601 video represented using 8 bits/pixel and 30 frames/s has a data rate of

$$\begin{aligned}
 & [(720 Y \text{ pixels/line}) \times (480 Y \text{ lines/frame}) + \\
 & (360 C_B \text{ pixels/line}) \times (480 C_B \text{ lines/frame}) + \\
 & (360 C_R \text{ pixels/line}) \times (480 C_R \text{ lines/frame})] \times (30 \text{ frames/s}) \times (8 \text{ bits/pixel}) \\
 & = (691\,200 \text{ pixels/frame}) \times (30 \text{ frames/s}) \times (8 \text{ bits/pixel}) \\
 & = 165\,888\,000 \text{ bits/s} \\
 & = 20\,736\,000 \text{ bytes/s} \\
 & \approx 70 \text{ Gbytes/hour}
 \end{aligned}$$

A typical compact disc (with approximately 650 Mbytes of storage) could store approximately 30 seconds of this video while a DVD [148] (with nearly 17 Gbytes capacity) can store approximately 15 minutes. The data rate of nearly 160 Mbits per second exceeds the capacity for many low cost, local-area networks such as 10 Mbits/s or 100 Mbits/s Ethernet. Clearly, processing uncompressed video is either expensive or impractical. The high data rates required for processing uncompressed video motivates the use of digital video compression.

## 2.4 Digital Video Compression

The large size of digital video signals is a challenge for storage, transmission, and processing. Digital video compression techniques allow a video signal to be represented more efficiently, which reduces the size and bandwidth of the video signal with a tradeoff in distortion. The use of compression is common in video applications, including distribution of prerecorded video [148], streaming video [149–151], video conferencing, digital television [152], and many others. Compression may be avoided in a relatively few applications (such as some video editing applications)

when distortion-free processing and processing costs have greater importance than video signal size or bandwidth. Video compression is an important consideration for video watermarking because of its prevalence in video applications, and that compression is an attack when performed on watermarked video.

Figure 2.8 shows the video compression framework. The objective of digital video compression is to encode or represent the digital video signal as efficiently and accurately as possible. Many approaches are used to obtain efficient representation, including quantization, prediction (including motion estimation and compensation), region and texture coding, model-based coding, and entropy coding [153–155].

The average data rate of the compressed video stream is the average number of bits the decoder reads and processes per unit time to reconstruct the video for real-time display. If the compressed video is transmitted over a communications network [149–151], the data rate determines the bandwidth required to transmit the video for real-time display (neglecting overhead and other features, such as error correction coding.) To obtain useful compressed data rates, most video compression techniques are lossy, where the reconstructed video signal  $V^*$  is not identical to the input video  $V$ . (If  $V^*$  is identical to  $V$ , the compression technique is lossless.) The difference between the input and the reconstructed video signals is the coding distortion. The coding distortion can cause artifacts to appear when the reconstructed video is displayed, such as blurring and blocking. The data rate and the amount of coding distortion of the reconstructed video are tradeoffs [132]. When the introduction of distortion is unavoidable, a goal of lossy video compression is to introduce distortion in a manner that minimizes its visibility.

#### **2.4.1 Video compression standards**

Video compression standards facilitate interoperability between systems and devices created by different manufacturers and avoid the problems associated with the use of incompatible, proprietary compression systems. By consensus, a video com-



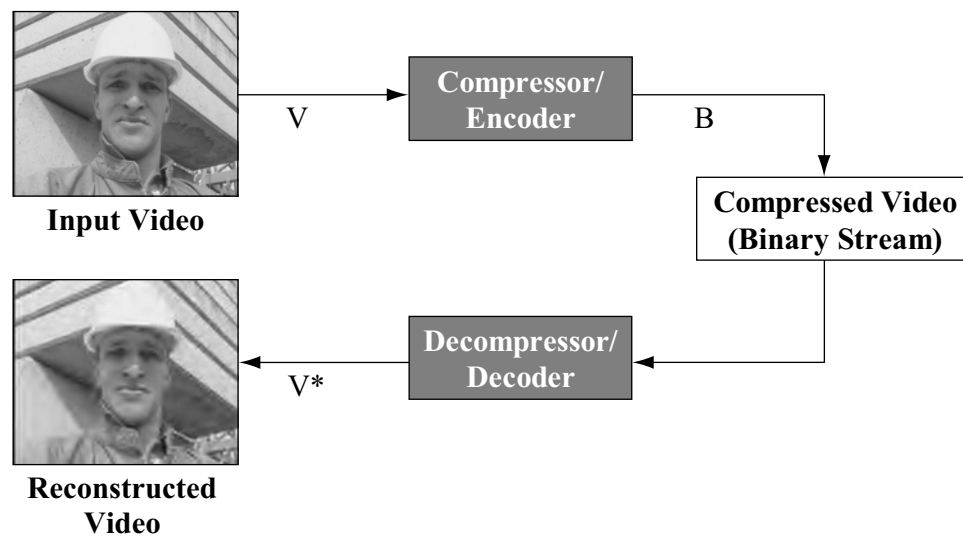


Fig. 2.8. Video compression

pression standard establishes the basic architecture and behavior of a decoder and defines the syntax of the compressed binary stream. By defining the behavior of a decoder, the standard ensures that the interpretation of a compressed stream is consistent regardless of the manner by which the stream was encoded. The behavior of encoders is not specified, and an encoder is free any means to generate a compliant binary stream.

Another feature of video compression standards is the use of annexes, levels, and profiles to define the coding limits and options available when using the standard. By restricting the coding parameters, a restricted or simplified version of the standard can be used in applications where implementing the full standard is expensive or unneeded. For example, in the MPEG-2 standard, profiles determine the subset of allowed coding and predictive features while levels determine the allowable range of parameters using in coding [156]. In the ITU standards (H.261, H.263), the fundamental decoding algorithm is described followed by many optional annexes that extend, modify, or replace sections of the basic algorithm. Some advanced coding features, such as scalable compression [149, 157–162] or error resilient compression [163–166], can be incorporated either as part of the main standard or as an optional annex or profile.

A brief list of some video compression standards follows.

- MPEG-1 [167]: Defined by the Motion Pictures Experts Group (MPEG) and finalized in 1993 for use in multimedia computing [168], the goal of MPEG-1 is to compress 4:2:0 CIF images at 30 frames/s (about 35 Mbits/s) to a rate of 1.5 Mbits/s, which is the data transfer rate of a single-speed CD-ROM drive. MPEG-1 employed motion compensation and estimation, the system layer for multiplexing audio and synchronization data, and rate control for video. MPEG-1 audio (particularly Layer III, or “mp3”) is very popular for compressing digital audio [53].

- H.261 [169]: A standard developed by the International Telecommunication Union, H.261 encoded 4:2:0 CIF and QCIF video for low data rates suitable for transmission through telephone lines. Interactivity and low coding latency are emphasized for videoconferencing and videophone applications. Unlike MPEG-1, H.261 defines only the coding of pictures and refers to other standards (for example, H.223 [170]) for multiplexing and audio.
- MPEG-2 [171]: A successor to MPEG-1 that supports more coding options, such as interlaced video and scalability [156]. MPEG-2 is currently used to store motion pictures on DVD discs (with data rates up to about 10 Mbits/s) and for the transmission of digital television in the United States, including high definition digital television (HDTV).
- H.263 [172] and its revisions: A successor to H.261 with optional annexes for more prediction modes, scalability, arithmetic coding, and other options [173]. Input picture formats other than 4:2:0 CIF and QCIF are supported. Like H.261, H.263 specifies only the coding of visual data and does not include multiplexing and audio.
- MPEG-4 [174]: A recent standard for coding video that supports shape coding, sprites, model-based coding, fine grain scalability [158], and many other features [175]. MPEG-4 introduces an object-oriented approach, where audio-video objects can be individually defined and then rendered together to form scenes. Figure 2.9 shows an example of object-oriented video coding. One of the features of the standard is to support enhanced user interactivity. The system layer and audio coding capabilities have also been expanded.
- H.264 [52]: The state-of-the-art video compression technique, also known as advanced video coding (AVC) or MPEG-4 Part 10, utilizes a variety of techniques to achieve considerable coding efficiency compared with the previous standards. These coding techniques include a new block transform [176] that

reduces “ringing” artifacts compared with the DCT, more refined spatial and temporal prediction, adaptive deblocking filter [177], arithmetic and context-based variable length codes for more efficient entropy coding [178], and many other refinements. H.264 has been observed to achieve a 50% reduction in rate when encoding at the same quality (distortion) as the other standards [52,179].

### 2.4.2 Hybrid predictive-transform coding

The video compression standards described in Section 2.4.1 are all implementations of a general coding framework known as hybrid predictive-transform coding, or simply hybrid coding [155]. Hybrid coding, shown in Figure 2.10, consists of four principal components: decorrelating transformation, quantization, prediction, and binary encoding. Transformation and quantization make compression easier by discarding information from the video. Prediction removes spatial and temporal redundancy in the video. Binary encoding removes redundancy in the compressed binary stream.

In hybrid compression, individual frames of the video can be encoded with or without temporal prediction. A frame that is coded without temporal prediction is known as an intra-frame, or I-frame. A frame that is coded using temporal prediction is an inter-frame. Temporal prediction uses one or more previously decoded frames, known as reference frames, as predictors for the inter-frame. The common types of inter-frames are predicted frames (P-frames) and bi-directionally predicted frames (B-frames.) Notably, the use of temporal prediction requires the encoder to contain the decoder and frame buffer, as shown in Figure 2.10. This structure allows previously reconstructed frames to be used as predictors for encoding subsequent frames. The feedback used for predictive coding is known the prediction loop. Hybrid coders can use multiple prediction loops to improve coding performance, for scalable coding, or for error resilience [162,180]. Figure 2.11 shows an example video

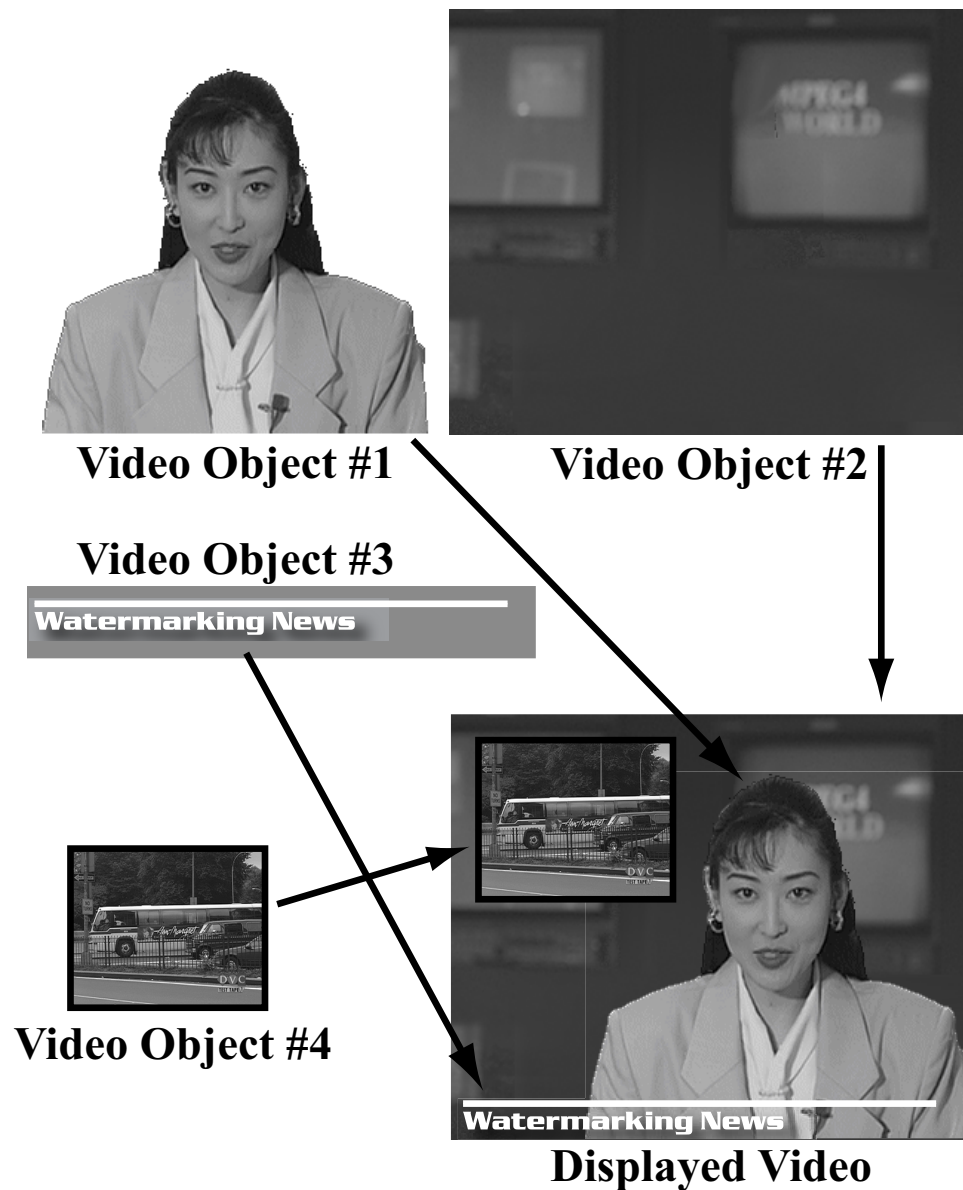


Fig. 2.9. Example of the object-oriented video compression supported by MPEG-4. Each video object is encoded as a separate elementary stream. The elementary streams for all the objects are multiplexed within the MPEG-4 compressed stream. An MPEG-4 decoder composes the scene from the individual objects for display.

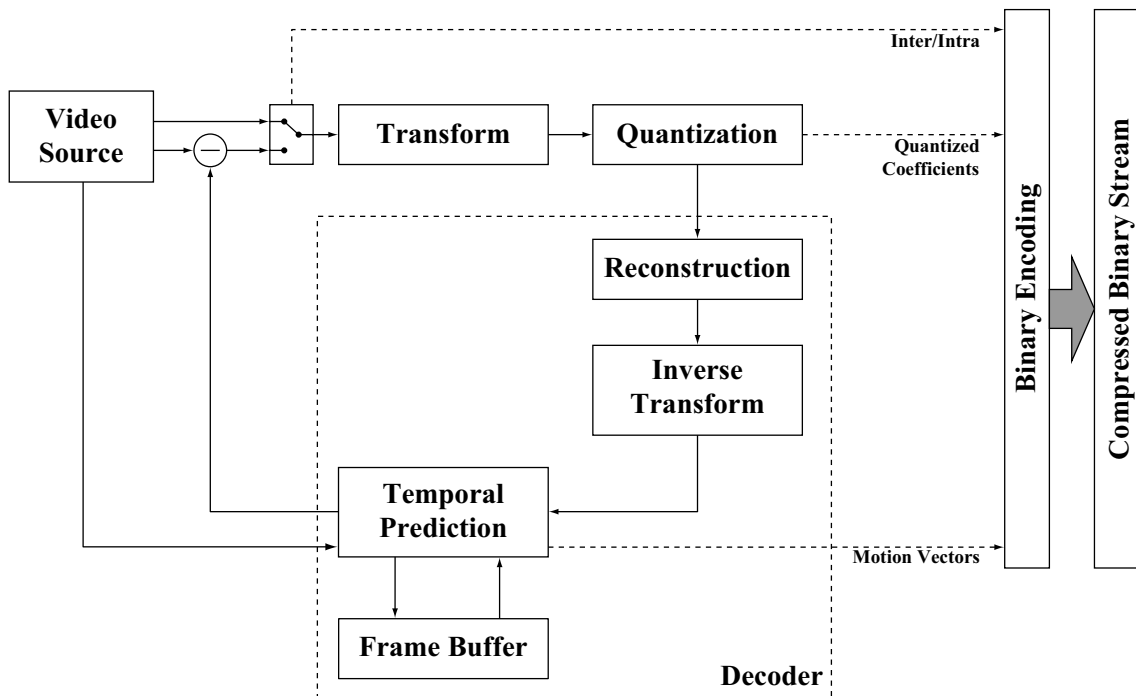


Fig. 2.10. Hybrid predictive-transform coding framework. The encoder contains a fully functional decoder within the prediction loop.

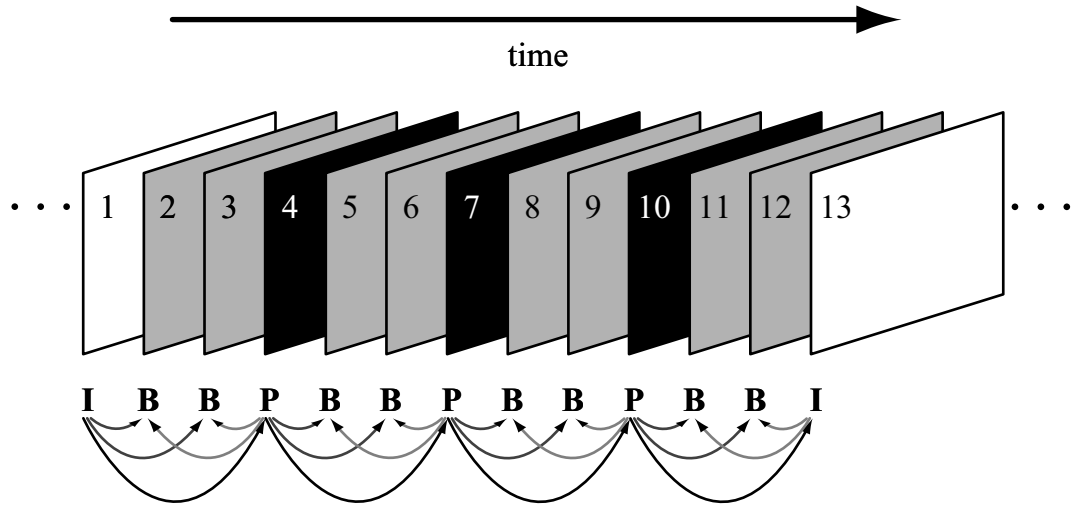


Fig. 2.11. Example of temporal prediction used in hybrid coding. I-frames are white, P-frames are black, and B-frames are gray. Frames are numbered to indicate display order. Arrows indicate temporal prediction, pointing from a reference frame to the predicted inter-frame.

encoded by a hybrid encoder. Arrows indicate temporal prediction, pointing from a reference frame to an inter-frame.

Intra-frame encoding uses a decorrelating transformation to transform the pixels of the frame to a set of transform coefficients, followed by quantization and binary encoding. The Karhounen-Loève Transform (KLT) is generally considered optimal or near-optimal for a decorrelating transformation [181–183] but the KLT has two disadvantages: High computational costs (computation of eigenvectors) and the necessity of obtaining or estimating the statistics (covariance) of the signal. Decorrelating transforms used for video compression include the Discrete Cosine Transform (DCT) [66,67,184], discrete wavelet transforms (DWT) [159,185–187], and [176]. After decorrelation, the transform coefficients are quantized. Quantization reduces the amount of information that is represented by the compressed video stream, but at the cost of introducing distortion into the video. The degree of quantization applied to the transform coefficients, or equivalently, the step size of the quantizer(s), are pa-

rameters by which the rate of the compressed video is adjusted. After quantization, the quantization indices are encoded in the compressed binary stream.

An inter-picture is temporally predicted using one or more reference frames. The processes involved in temporal prediction are motion estimation and motion compensation. In motion estimation, the reference frames are searched to locate the best predictors (closest matches) for the inter-frame. Typically, the inter-frame is partitioned into blocks and the search is performed on each block. The predictors are encoded in the compressed stream as a set of motion vectors. How the search is performed is dependent on the implementation of the encoder. Motion estimation is computationally expensive because a search is required for each block of the inter-frame.<sup>6</sup> Motion compensation is the application of the motion vector information to predict the inter-frame from its reference pictures. Motion compensation is performed by a decoder and does not require search. Therefore, motion compensation does not have the large computational cost of motion estimation.

After temporal prediction, the prediction residual (known as the predictive error frame, or PEF) is encoded in a similar process as an intra-frame. Summarizing, the compressed binary stream contains the following elements:

1. For each intra-frame: The quantized transform coefficients of the image.
2. For each inter-frame: Motion vectors describing temporal prediction, and quantized transform coefficients of the PEF.
3. Headers and other information that is needed by a decoder to reconstruct the video.

Decoding the binary stream is conceptually straightforward. The decoder parses the stream and extracts the encoded data for each frame. For an intra-frame, the decoder obtains the quantization indices from the compressed stream, reconstructs the transform coefficients, and then applies the inverse transform to reconstruct the

---

<sup>6</sup>There is interest in reducing the amount of computation at the encoding device, or distributed encoding [162, 188].



intra-frame. For an inter-frame, the decoder retrieves each of the reference frames from the frame buffer. The motion vectors are obtained from the stream and motion compensation is applied to form the predictor image. The decoder decodes the PEF (similar to an intra-frame) and then adds the PEF to the predictor image to obtain the reconstructed inter-frame. Reconstructed frames are stored in the frame buffer for temporal prediction or display.

## 2.5 Image and Video Watermarking

This section summarizes the methodologies and approaches that have been used for the watermarking of digital image and video signals. The number of proposed watermarking techniques in the published literature is staggering, which reflects the immense interest in watermarking and watermarking applications. While it is not possible (nor useful) to review the large number of techniques in exhaustive detail, watermarking techniques share some common principles. The focus of this section will be on these common principles, and the reader is encouraged to review tutorials and overviews [27–33] for additional breadth.

### 2.5.1 Image watermarking

Early watermarking techniques [189–197] generated the watermark signal as a pseudo-random noise-like pattern. A variety of random number distributions were proposed for generating the watermark samples. Some watermarks were produced using a sequence of Bernoulli-distributed samples, such as bi-polar ( $-1$  and  $1$ ) or binary ( $0$  and  $1$ ) sequences. Other watermarks were white or colored sequences of uniform or Gaussian distributed samples, but generally any probability distribution could be used. The embedding key was often the seed to the pseudo-random number generator (see Section 2.7.1). Many techniques inserted the watermark by additive embedding, or by replacing the least-significant bits (LSB) of the pixel values of the original image with the watermark. These techniques usually relied on the low

amplitude of the watermark signal for perceptual transparency and did not explicitly use visual modeling. Watermark detection is usually based on correlation.

Perceptual models were proposed to improve the perceptual transparency and robustness of watermarking techniques [32, 34, 198–203]. A perceptual model predicts the amount of distortion that can be inserted into (specific regions of) an image without detrimentally impacting the overall perceptual quality. Applied to watermarking, a perceptual model allows the embedder to increase the amplitude or power of the watermark in regions where watermark embedding is not likely noticed. With or without attacks, watermark detection is easier with increased embedding power and watermark robustness is improved. Conversely, the watermark is attenuated (or not embedded at all) in regions where watermark embedding is likely to be noticed. Perceptual models vary in complexity and may account for many different properties of the human visual system, including frequency sensitivity, luminance sensitivity, and contrast masking [29, 34, 198, 204]. Adjusting the power of an embedded watermark by using a perceptual model is also known as perceptual shaping. A disadvantage of perceptual shaping is that perceptual modeling increases the complexity, and thus the cost, of watermark embedding. Perceptual models may also be used to attack watermarks [205].

Robust watermarking became more effective by the recognition that robustness required inserting the watermark into “perceptually significant” regions of an image [61]. A watermark embedded only in perceptually insignificant regions could be removed by attack without causing any appreciable decrease in the quality of the attacked image. Many early watermarking techniques embedded the watermark solely in perceptually insignificant regions of the image, which limited their robustness.

While embedding in perceptually significant regions was recognized, there has been debate in how or where the watermark should be embedded. Some techniques have suggested embedding within certain spectral components of an image [206–210], such as low-frequencies, middle-frequencies, or high-frequencies. Su and Girod showed that a spectrally adaptive watermark is most robust against Wiener filtering

and estimation attacks [57, 211]. Some techniques proposed embedding the watermark in the spatial domain [212, 213], but many techniques have suggested representing the image as transform coefficients and embedding the watermark in a transform domain. Such transforms include the DCT [32, 43, 61, 203, 207, 214], Discrete Fourier Transform (DFT) [105, 210], DWT [32, 199, 208, 215–221], and others [200, 222–225]. Watermarking techniques may take advantage of the (special) properties of a particular transformation to improve performance, such as [200, 223, 224, 226]. For watermarking color images, many techniques watermarked only the luminance and do not watermark the chrominance components. However, some techniques have suggested watermarking in the chrominance [214, 226–228] of a color image.

Many robust watermarking techniques [60, 61, 229, 230] generate and embed the watermark using principles from spread-spectrum communications [231]. In classical spread-spectrum, a (spectrally) narrow-band input signal is encoded or modulated to produce a wide-band signal, and the wide-band signal is transmitted through a noisy or lossy channel. The process of producing the wide-band signal from the input signal is known as spreading. While transmitting the wide-band signal requires more bandwidth, spreading reduces the effect of noise and interference that may be introduced in the channel (for example, from jamming). There are also other benefits of spreading [231]. The receiver obtains the wide-band signal and recovers the input signal through decoding.

The benefits of spread-spectrum are useful in robust watermarking. Spread-spectrum watermarks treat the payload as the narrow-band signal and the watermark itself as the wide-band signal. The robustness of the watermark arises from the resilience of spread-spectrum encoding against noise. Removal attacks are analogous to jamming or a noisy channel. Interference suppression is useful for watermark detection, which may be subject to host-signal and inter-watermark interference. Spreading is often accomplished by encoding individual bits of the payload with pseudo-random sequences, which is a method known as direct sequence spread-spectrum (DSSS). Pseudo-random sequences have desirable auto-correlation

and cross-correlation properties [232]. In DSSS, the receiver recovers the payload by matched filtering (correlation).

Recent informed embedding techniques explicitly use information available at the embedder to design more effective watermarks [28, 56, 58, 59, 64, 122, 123]. Such information include (characteristics of) the original image, the embedding method, the detection method, and potential attacks. While earlier techniques often generated and embedded the watermark in a fixed manner, informed embedders adapt the structure of the watermark for improved performance. For example, host-signal interference may be reduced [56]. An informed embedder has the potential to adapt the watermark to anticipate attacks [123], and generate image- and attack-dependent watermarks.

### 2.5.2 Video watermarking

A naïve way of extending still image watermarking techniques to video is to treat the video as an ordered sequence of still images, which are then watermarked individually. Some techniques [61, 212, 233–235] suggest this approach. These techniques (1) embed the same watermark signal into each frame, (2) embed different (uncorrelated) watermarks into each frame, or (3) do not precisely specify how the watermark is generated temporally and merely state “the (still-image) technique is applied to each frame of the video.” These frame-by-frame techniques generate and embed the watermark into each frame independently, without regard to temporal structure of the video.

Unfortunately, neglecting the temporal structure of video leaves the watermark vulnerable to collusion attacks. For many videos, the frames that are displayed temporally close are usually correlated while frames that are displayed temporally distant are less correlated or uncorrelated. Embedding the same watermark in all frames of a sufficiently lengthy video results in a large number of uncorrelated images that are watermarked by a common watermark. With a sufficient number of watermarked

frames, a good estimate of the watermark can be obtained by averaging. Conversely, embedding uncorrelated watermarks into correlated frames leaves the watermark vulnerable to removal by averaging. These vulnerabilities were mentioned in [89] and examined more closely in [82, 236].

Collusion attacks allude that there is more to video watermarking than merely the repeated exercise of still image watermarking. Video and still image watermarking are now compared.

First, applications often process video in ways that are either impossible (because still images are not temporally varying signals) or atypical in still image applications. This processing includes common video editing operations such as interlacing and deinterlacing [139, 237], frame-rate conversion (including 3:2 pulldown [139]), temporal cropping, and aspect ratio conversion (including pan-and-scan [148]). Thus, video watermarks may require robustness against attacks (not only collusion attacks) that do not occur for still image watermarks.

Second, computational costs of watermark embedding and detection are generally more significant issues for video than still images. One reason for the increased emphasis on computational cost is that video data is often much larger than still image data. As a result, processing a single video signal is more costly than processing a still image. Another reason is that some video applications necessitate real-time processing, which would include watermark embedding or detection.

Third, video watermark detectors may not have the opportunity to examine or process the entire input signal. Even when the watermark detector has (physical) access to the entire video, the size of the video data makes reading and processing the video time consuming. Applications and users often require a response from the detector much before an entire video can be examined by the detector. And even when processing time is not constrained, buffering entire video signals is expensive. In some applications, for example broadcast monitoring of digital television, the video is regarded as a very long signal and the concept of “entire video” is not well-

defined. In contrast, watermark detectors for still images are often able to examine an entire image before deciding the detection result.

When an application does not or cannot provide the entire video stream to the watermark detector, the detector is constrained to processing sections of the video at a time. These sections may be as small as single frames of the video, perhaps even portions of a single video frame. Furthermore, any portion of the video may be provided to the detector and not necessarily what a user would consider the “beginning” of the video. This has implications for video watermark synchronization and will be elaborated in Section 2.6.

Lastly, designing perceptual models for video watermarking is more challenging. A perceptual model for video should consider both temporal and spatial properties [34,112] of the human visual system. Some types of distortion produce perceptual phenomena that is only visible for temporally varying signals, such as flickering and shimmering (mentioned in Section 2.2.1). Modeling temporal properties of visual perception requires a more complex and costly perceptual model than one for still images. Conversely, cost limitations imposed by applications constrain or prevent the use of perceptual models which require expensive analysis.

Video watermarking techniques have used three approaches for watermark embedding: uncompressed embedding, compressed domain embedding, and joint compression watermarking. These approaches differ in the interaction between watermark embedding and video compression.

In uncompressed embedding, both the original video and the watermarked video are uncompressed. The original video is viewed as a spatially- and temporally-varying signal of pixels. The pixel values are modified to insert the watermark and produce the watermarked video. Similar to still image watermarking, the watermark may be embedded in a transform domain [238–241] or in the spatial domain [38, 242]. The naïve frame-by-frame techniques mentioned previously are examples of uncompressed embedding. Uncompressed embedding is agnostic to and separate from video compression.

While uncompressed embedding is suitable when compression is not used, compressing the watermarked video is a removal attack with respect to watermarking. An application that requires the watermarked video to be compressed is forced to attack the watermarked video. Therefore, the watermark is embedded with excess robustness such as to possess the degree of robustness needed by the application after compression. Embedding with excess robustness incurs a performance cost, such as increased watermark visibility or decreased capacity.

In compressed domain embedding, the original video is provided to the embedder as a compressed stream. The embedder partially decodes the stream to expose elements of the compressed video, such as transform coefficients [104, 242–244], motion vectors [245, 246], or video objects [247–250]. The elements of the partially decoded video are modified to insert the watermark, and then reassembled to form the compressed watermarked video stream. The reassembly step retains the coding decisions that were used to encode the original video whenever possible and requires much less computational effort than compression.<sup>7</sup> The watermark embedder ensures that the watermarked video is a valid compressed video stream, which can be decoded by using a standard decoder. These steps are summarized in Figure 2.12.

The benefits and disadvantages of compressed domain watermarking arise from processing a compressed video stream. In contrast with uncompressed embedding, watermarking with excess robustness is unnecessary because the watermarked video is already compressed and no further compression is needed. In addition, the compressed video stream contains information, such as prediction and quantization parameters, which allows informed decisions to improve robustness, transparency, or other performance criteria. In uncompressed embedding, the embedder processes the video prior to compression, hence compression information is generally not available to the embedder.

---

<sup>7</sup>Because reassembly does not involve search, the watermarked video may be sub-optimally represented. The loss of coding performance caused by using reassembly instead of re-compression after watermark insertion has not been explored.

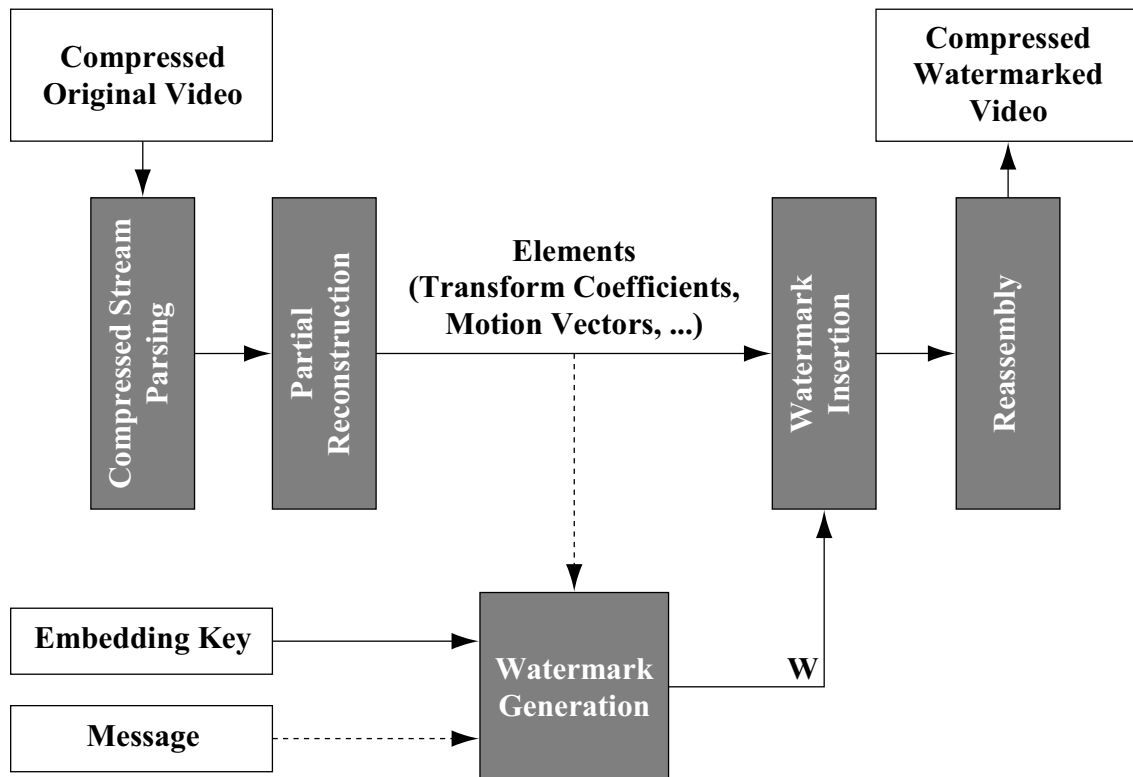


Fig. 2.12. Compressed domain watermark embedding. The original compressed video is parsed and partially decoded to expose elements of the compressed video data. Some of the elements are modified to insert the watermark, and then the compressed data is reassembled to obtain the watermarked video.



A compressed domain embedder is specific to the compression technique used to encode the original video. This specificity arises because the syntax of the compressed video stream is strongly dependent on the video compression technique. As a result, changing the compression technique used by an application necessitates changing the watermark embedder. In addition, the performance of the watermark embedder is dependent on the implementation of the compressor and the compression parameters. Some coding decisions (made by the compressor) generate compressed streams that are more easily watermarked. For example, an embedding technique which inserts the watermark in the motion vectors will be more effective when watermarking video streams containing many inter-frames than streams containing many intra-frames (because intra-frames lack motion vectors and cannot be watermarked by the technique).

Lastly, compressed domain embedding introduces issues of drift compensation and data rate control [242,243]. Drift occurs when prediction is used in compression. Modifying an element in the compressed stream (such as to insert the watermark) affects the reconstruction of all other elements which use the modified element as a predictor. The difference between the reconstruction of the predicted elements using the original and modified predictors is drift. Unless the drift is compensated, the accumulation of drift results in very noticeable artifacts in the watermarked video. In addition to drift compensation, the data rate of the watermarked video must be controlled to within application-specified limits. If no measures are taken to control the data rate, watermark insertion will usually cause an increase in the rate of the watermarked video (compared to the rate of the original video.) The increase occurs because watermarks are generally noise-like signals that are difficult to represent efficiently.

A joint compression watermarking technique [251] accepts uncompressed original video and produces a compressed watermarked video stream by combining compression and watermark embedding together. The combination should be more tightly coupled than compression followed by watermark embedding, since the latter would

be the same as compressed domain embedding. For example, the watermark embedder may be within the prediction loop of a hybrid coder. The watermark embedder and compressor are able to influence each other and performance improvement may be possible over compressed domain embedding. However, this approach is largely unexplored and rigorous performance comparisons have not been made to date.

A brief overview of notable video watermarking techniques follows.

Hartung [3, 242] applies (direct sequence) spread-spectrum techniques towards video watermarking for both uncompressed embedding and compressed-domain embedding. For uncompressed embedding, the watermark is inserted into the pixels of the video. For compressed-domain embedding, the watermark is inserted into the DCT coefficients of MPEG-2 video streams with drift compensation. Data rate is controlled by watermarking only non-zero DCT coefficients, and then, only if watermark insertion does not increase the rate. Watermark detection is correlation-based with optional pre-filtering. The technique was evaluated for high rate (8 Mbits/s) video. Alattar [243] extends the spread-spectrum approach for low rate MPEG-4 video, with drift compensation, a rate control method that is more suitable for low rate video, and a simple model for perceptual shaping. The MPEG-4 watermarking technique also embeds templates for spatial synchronization; Templates will be discussed later in Section 2.6.3.

Langelaar [36] has proposed two compressed-domain embedding techniques. The first technique [244] embeds the watermark by modifying the variable length binary codes of DCT coefficients encoded in MPEG-2 video. This technique is similar in principle to least-significant-bit embedding used in early still-image watermarks. The second technique [104] is known as Differential Energy Watermarking (DEW). In DEW, an intra-frame is partitioned into sets using the embedding key, with each set consisting of two subsets containing the same number of DCT blocks. Each set expresses a single payload bit, depending on which of the two subsets contains more energy in the reconstructed DCT coefficients. When necessary, DCT coefficients are zeroed within a subset such that the desired payload bit is expressed. Changing

a non-zero DCT coefficient to zero reduces the rate, thus watermark embedding is guaranteed to not increase the rate. The detector compares the energies of the subsets and does not involve correlation. While the original DEW technique only watermarked the intra-frames, DEW has been extended for embedding into inter-frames [252].

Swanson [238] describes a technique whereby the watermark is inserted into scenes using a three-dimensional spatiotemporal wavelet transform and perceptual shaping. Because the wavelet transform is performed across an entire scene, this technique has relatively high memory and computational costs. An advantage of this technique is that the embedded watermark has some robustness when frames of the watermarked video are lost or processed.

Deguillaume [239] describes a technique which embeds the watermark in the mid-frequencies of the three-dimensional spatiotemporal Discrete Fourier Transform (3D-DFT). The watermark is inserted into fixed-sized groups of frames and this technique does not require entire scenes to be processed like [238]. Watermark generation using the 3D-DFT produces a locally correlated structure in the spatial domain, which provides some benefits against local frame averaging or collusion attack. This technique also embeds a template for robustness against synchronization attacks.

Kalker [38] proposes a watermark for broadcast monitoring, known as Just Another Watermarking System (JAWS). The watermark is constructed as a tiled pseudo-random pattern and additively inserted in the spatial domain of each video frame. Detection uses the phase of the DFT, which takes advantage of the tiled structure to allow efficient watermark detection even if the frame has been spatially shifted.

While most video watermarking efforts have focused on embedding the watermark in the visual portion of video, videos may also contain auditory signals that can be involved in watermarking. For example, a watermark may be embedded into both the visual and auditory portions of the video [253]. Auditory information can be used to generate the watermark which is inserted into the visual portion of the video, and

conversely, the visual portion of the video can be used to generate a watermark that is embedded in the audio portion.

## 2.6 Synchronization

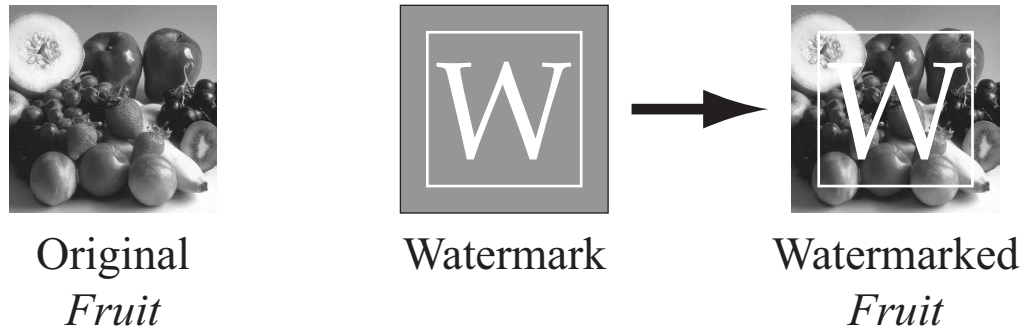
Watermark detector synchronization, or simply synchronization, is recognized as a significant challenge in blind robust watermark detection. Most watermark detectors will fail to detect an embedded watermark if synchronization cannot be achieved. In this section, the concepts of synchronization and synchronization attacks are defined and current approaches for addressing synchronization are reviewed.

Watermark insertion introduces a correspondence between the coordinates of the embedded watermark and the coordinates of the watermarked signal. This correspondence arises because the watermarked signal  $Y$  is defined as a function of the original signal  $X$  and the watermark  $W$ , causing individual samples of  $W$  to be inserted into samples (pixels or transform coefficients) of  $Y$ . For example, re-writing the additive embedding equation (2.1) to explicitly indicate the samples being added, and defining the coordinate vector  $\mathbf{v}$  similarly to (2.10), then

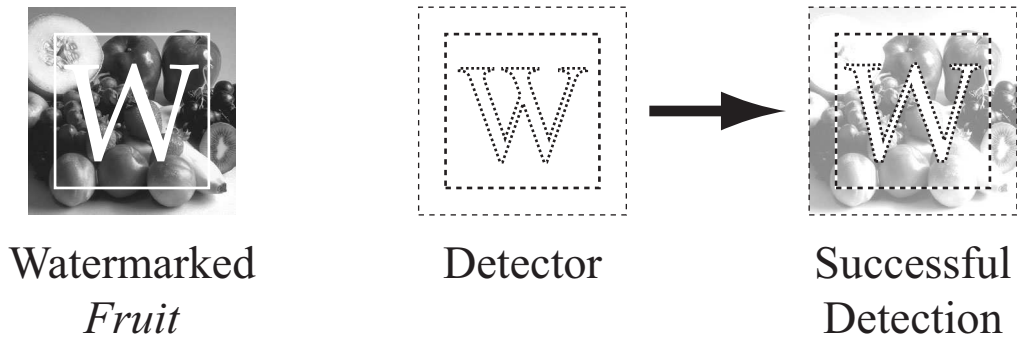
$$Y(\mathbf{v}) = X(\mathbf{v}) + \gamma W(\mathbf{v}) \quad (2.12)$$

and the correspondence is that the  $\mathbf{v}$ -th watermark sample  $W(\mathbf{v})$  is embedded into the  $\mathbf{v}$ -th sample of  $Y$ . This correspondence extends to both the spatial and temporal dimensions for video watermarks. Conceptually, the embedded watermark is “positioned” in the watermarked signal in accordance to the watermark insertion technique. Figure 2.13(a) illustrates this correspondence for a watermark embedded using a hypothetical image watermarking technique.

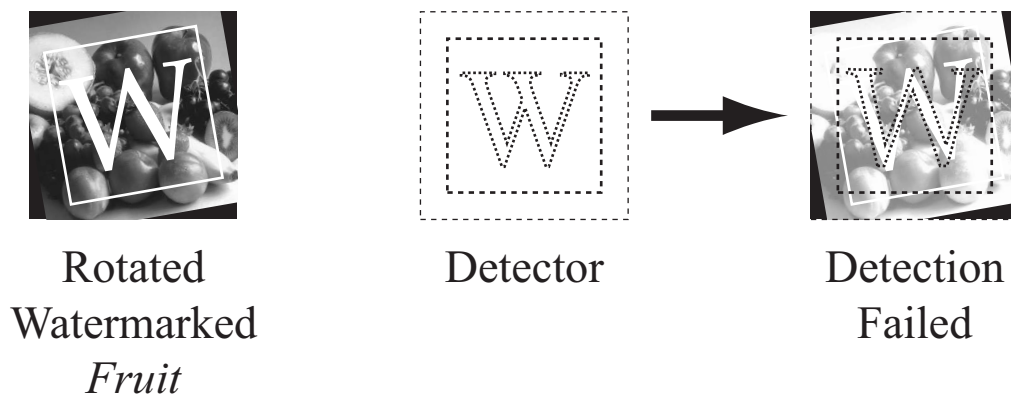
The watermark detector is designed with knowledge of the embedding technique, and thus the position of the embedded watermark. When the watermarked signal is provided to the detector ( $Z = Y$ ) with the appropriate detection key, as shown in Figure 2.13(b), the detector successfully detects the watermark. (Assume for this



(a) Watermark embedding using hypothetical watermarking technique. Watermark is normally invisible but shown here to emphasize its location and orientation when embedded in the watermarked image.



(b) Watermark detection without attack. Detector examines image and discovers that the watermark is present at its expected position. Watermark is detected.



(c) Watermark detection with synchronization attack. Rotating the watermarked image causes the watermark to be rotated. Detector examines the rotated image, but the watermark has been rotated and cannot be detected in its original position.

Fig. 2.13. Watermark embedding, detection and synchronization

discussion that a miss does not occur due to interference or other signal detection issues.)

Now suppose that a synchronization attack is applied to the watermarked signal, such as the rotation of the watermarked image shown in Figure 2.13(c). A coordinate transformation applied to the watermarked signal transforms the embedded watermark in an identical manner as the watermarked signal in the spatial domain. For most signals, the position of the embedded watermark relative to the attacked signal  $\hat{Y}$  is changed and no longer identical to the position of the watermark prior to the attack. Some watermarks are invariant to particular coordinate transformations, which will be discussed later in Section 2.6.2. When  $Z = \hat{Y}$  is provided to the watermark detector, the detector will fail to detect the watermark in its original position. The attacked signal is watermarked, but the detector misses unless it can determine or deduce the position of the watermark in the attacked signal.

Synchronization is the process of identifying the correspondence between the spatial and temporal coordinates of a watermarked (and possibly attacked) signal and that of an embedded watermark. An intuitive description of synchronization is “finding the watermark.” As the prior example shows, the watermark is inserted at some position in the watermarked signal. If the watermarked signal is not attacked, then synchronization is trivial because of the detector’s knowledge of the embedding technique. However, when the watermarked signal is attacked, the position of the embedded watermark may change. With most watermarking techniques, the watermark detector misses because successful detection is possible only when examining the watermark where it resides after the attack. Even spread-spectrum watermarks, which have demonstrated robustness against signal processing attacks and addition of noise, are vulnerable to detection miss under a synchronization attack. The challenge is to determine how the watermark has been relocated (which is synchronization), or to design watermarks that are less sensitive to synchronization attacks [254].

While the possibility of malicious attacks alone provides sufficient motivation to address synchronization, synchronization is generally an issue for robust watermark detection even in the absence of malicious attacks. Some applications process signals in ways that may disturb the positions of embedded watermarks. For example, video watermarked for broadcast monitoring may be subjected to editing that is typical for broadcast television video, including aspect ratio conversion, frame rate conversion, (spatial and temporal) cropping, and video splicing. Other applications do not deliberately process a watermarked signal but nonetheless require robust watermark detection. For example, when a watermark is embedded to trace illicit copies of a motion picture made by theater recording [255], or so-called “camcorder piracy,” the position of the watermark is changed in an illicit copy because the recorder is not perfectly aligned with the movie display. Similarly, when a watermarked image is printed and then scanned [46, 70], the position of the watermark in the scanned copy will be changed. Damage to the watermarked signal may also cause the watermark detector to lose synchronization. One example where signal damage may occur is network transmission in video streaming and broadcast applications [2, 149–151]. Network congestion and noise may even cause the signal to be entirely lost for an indeterminate time.

Video watermarking applications have additional reasons to be concerned with synchronization. As mentioned in Section 2.5.2, many video watermark detectors do not have the opportunity to examine the entire input video, but only sections of the video at a time. These detectors must synchronize when the input signal first becomes available, which is a process known as initial synchronization. The significant challenge for initial synchronization is that any portion of a video may be provided to the detector. A robust watermark detector generally cannot rely on observing a specific portion of the video (such as the “beginning” of the video) because (1) that portion of the video may not be provided to the detector, and (2) relying on observing a specific portion of the video for synchronization is a vulnerability. Specifically, the detector would be vulnerable to an attack which crops that portion of the video.

Once initial synchronization is successful, the detector can continue to detect the watermark as the input video is available until an attack or some other event (such as loss of the input signal) causes synchronization to be lost. After synchronization loss, the detector must resynchronize with the input to resume watermark detection.

Synchronization is also an issue for watermarking techniques which insert the watermark into individual video objects [247–250]. For example, each video object of Figure 2.9 may be individually watermarked. To detect the watermarks, the watermark detector must locate the position of each watermarked object in the video. This is a synchronization problem [256]. Information in the compressed stream may identify the position of the video objects, which would make synchronization relatively easy. However, such information is lost if the video is uncompressed, or transcoded into a new compressed format which does not support video object coding.

### 2.6.1 Synchronization attacks

Synchronization attacks are part of the class of detection-disabling attacks where the watermark is obfuscated by relocation. The embedded watermark is moved, but not destroyed, by a synchronization attack. Like other attacks, synchronization attacks can occur from innocuous or malicious signal processing. When synchronization attacks occur maliciously, the objective of the attacker is to cause the watermark detector to miss. Often, causing a miss is considerably easier than destroying or removing an embedded watermark.

Most synchronization attacks are coordinate transformations applied to a watermarked signal. Coordinate transformations relocate the embedded watermark, forcing the watermark detector to confront a more challenging synchronization problem. The processing involved in a coordinate transformation is a (re-)mapping followed by interpolation, which is generally easy to accomplish. No knowledge of the watermark structure or keys is required. Furthermore, a coordinate transformation



does not necessarily affect the watermarked signal in a dramatic or obvious manner. Subtle coordinate transformations are often sufficient to cause a watermark detector to miss without synchronization, although whether a specific transformation is sufficient or not depends on the transformation and the watermarking technique.

Spatial synchronization attacks, or geometric attacks, are transformations that change the spatial position of the pixels of a watermarked image or video frame. These transformations include, but are not limited to, rotation, uniform and non-uniform scaling, translation (or shifting), reflection (or flipping), perspective transformation, and warping. Table 2.4 shows examples of these transformations applied to the *Fruit* image of Figure 2.3. For each transformation, the grid (left column) and the left hand image (middle column) have been transformed identically. The right hand image (rightmost column) has been transformed in a more subtle fashion. Stated with emphasis, the transformations applied to the right-hand images are generally sufficient to cause many watermark detectors to miss without synchronization.

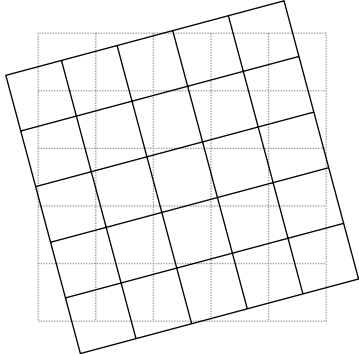


Temporal synchronization attacks are transformations that change the temporal structure of a watermarked video signal. These transformations include frame dropping, insertion, transposition, and averaging. A frame drop occurs when a frame is removed from the video. Multiple frames may be dropped from the video, at regular intervals or at random. Temporal cropping or removal of a section of video can result in many successive frames being dropped. Temporal decimation (by factor  $\lambda$ ) is a special case of frame dropping where a single frame is retained for every  $\lambda$  frames of the video while all other frames are dropped. Frame insertion is the insertion or splicing of one or more arbitrary frames into the video. A special case of frame insertion is temporal upsampling, where each frame of the video is repeated  $\lambda$  times. Frame transposition is the interchanging of two or more frames of the video, which changes the order in which those frames are displayed. Frame transposition does not necessarily involve adjacent frames. Frame averaging uses a moving window of  $\lambda$  frames, where the value of each pixel in the attacked video is obtained by the

arithmetic mean of the pixel values at the corresponding spatial location of each frame in the window.

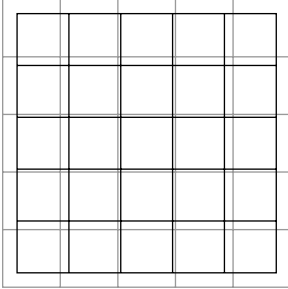


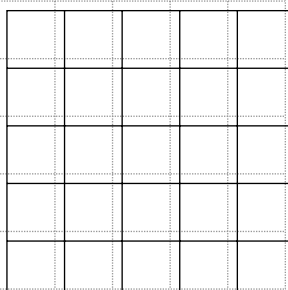


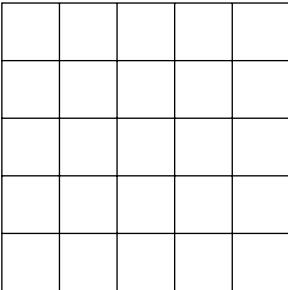


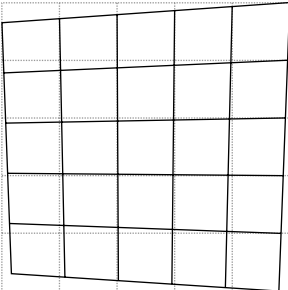
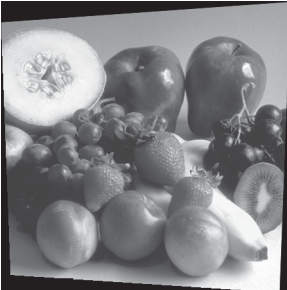

Temporal synchronization attacks also include more general processes of temporal filtering and temporal warping. Temporal filtering is the application of a filter with impulse response involving the temporal dimension (i.e. multiple video frames.) Frame averaging is an example of finite impulse response (FIR) temporal filtering, where each tap has weight  $1/\lambda$ . Temporal filtering may be combined with temporal decimation or upsampling as part of frame rate conversion, which is conceptually identical to sampling rate conversion [137]. Temporal warping is temporal filtering with a spatiotemporally variant filter.

Aside from coordinate transformations, synchronization attacks may also target the synchronization mechanism used by the watermark detector. These synchronization attacks do not necessarily cause synchronization loss by themselves, but they disable or inhibit the detector's ability to synchronize with the attacked signal. These attacks may be combined with coordinate transformations to challenge the watermark detector. Examples of these attacks are discussed in Section 2.6.3.

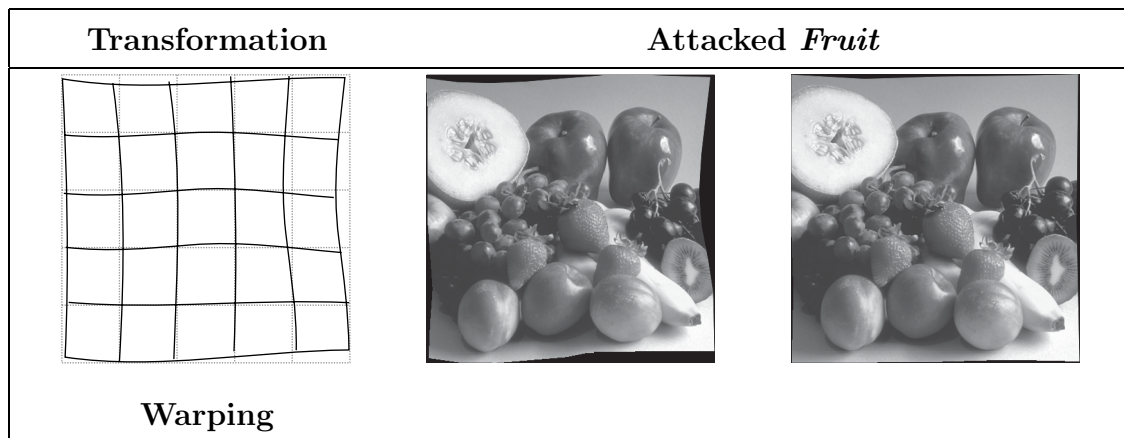
Table 2.4: Examples of spatial synchronization attacks

Transformation	Attacked <i>Fruit</i>	
		
<b>Rotation</b>	15° Rotation	0.5° Rotation

*Continued on next page*

Transformation	<i>Attacked Fruit</i>	
		
<b>Scaling</b>	10% Reduction	1% Reduction
		
<b>Translation</b>	15 pixel shift	1 pixel shift
		
<b>Reflection</b>	Horizontal axis	Vertical axis
		
<b>Perspective</b>		

*Continued on next page*



### 2.6.2 Synchronization insensitivity and invariant transformations

One way to address synchronization issues is to design watermarks that are less sensitive to synchronization attacks. Invariant domain watermarking is an approach to render some synchronization attacks ineffective. Many image and video watermarking techniques embed the watermark in a transform domain, by applying a signal transform<sup>8</sup> to represent the original signal as coefficients, and then inserting the watermark into the coefficients. The inverse signal transform is applied to obtain the watermarked signal in the spatial domain. Some signal transforms have the property that certain coordinate transformations applied to the signal in the spatial domain do not change or affect the coefficients of the transform domain. These signal transforms are said to be invariant to the coordinate transformations.

Invariance properties allow the watermark detector to bypass synchronization. When the watermark is embedded in a domain invariant to particular coordinate transformations, those transformations in the spatial domain do not relocate the watermark. Because the watermark is not relocated, the coordinate transformation does not desynchronize the watermark detector. Watermarks that are embedded in

<sup>8</sup>The word “transform” alone may be confusing in this section. “Signal transform” is used for the concept of transforms like the DFT and DCT, to avoid confusion with “coordinate transform.” Similarly, “transform coefficients” will be referred to as “signal transform coefficients” or shortened to simply “coefficients.”

transform domains specifically for their invariance properties are known as invariant domain watermarks.

An example of invariance is the magnitude of the DFT with respect to spatial translation or shifting. A well-known property of the DFT is that (cyclic) shifting in the spatial domain results in only a phase change in the DFT [135, 137, 138]. A watermark that is embedded in the magnitude of the DFT (and not the phase) is invariant to shifts in the spatial domain.

Several watermarking techniques take advantage of the Fourier-Mellin transform [223, 224]. To obtain the Fourier-Mellin transform, a two-dimensional signal is first transformed using the log-polar mapping (LPM) and then the Fourier Transform is applied to the LPM-transformed image. The LPM is a mapping  $(x, y) \rightarrow (\mu, \theta)$  from each point in Cartesian  $(x, y)$  space to a unique point in log-polar  $(\mu, \theta)$  space, analogous to transforming from a Cartesian coordinate system to polar coordinates. Specifically,

$$\mu = \log \sqrt{x^2 + y^2} \quad (2.13)$$

$$\theta = \arctan\left(\frac{y}{x}\right) \quad (2.14)$$

and the inverse LPM is

$$x = e^\mu \cos(\theta) \quad (2.15)$$

$$y = e^\mu \sin(\theta) \quad (2.16)$$

The interesting property of the LPM is that uniform scaling and rotation in Cartesian space are mapped to shifts in the log-polar space. For example, uniform scaling by scale factor  $\rho$  maps the point  $(x, y)$  to  $(\rho x, \rho y)$  in Cartesian space, or

$$\mu' = \log \sqrt{(\rho x)^2 + (\rho y)^2} = \log \left[ \rho \sqrt{x^2 + y^2} \right] = \log \sqrt{x^2 + y^2} + \log \rho \quad (2.17)$$

$$\theta' = \arctan\left(\frac{\rho y}{\rho x}\right) = \arctan\left(\frac{y}{x}\right) \quad (2.18)$$

in log-polar space. Comparing (2.17) and (2.18) with (2.13) and (2.14), it is seen that uniform scaling in Cartesian space is a shift (by  $\log \rho$ ) in log-polar space, or

$$(\rho x, \rho y) \rightarrow (\mu + \log \rho, \theta) \quad (2.19)$$

Rotation by  $\eta$  (counterclockwise) in Cartesian space maps the point  $(x, y)$  to  $(x \cos \eta - y \sin \eta, x \sin \eta + y \cos \eta)$ . In log-polar space, this is

$$\begin{aligned} \mu'' &= \log \sqrt{(x \cos \eta - y \sin \eta)^2 + (x \sin \eta + y \cos \eta)^2} \\ &= \log \sqrt{x^2 [\sin^2 \eta + \cos^2 \eta] + y^2 [\sin^2 \eta + \cos^2 \eta]} \end{aligned} \quad (2.20)$$

$$= \log \sqrt{x^2 + y^2} \quad (2.21)$$

$$\begin{aligned} \theta'' &= \arctan \left[ \frac{(x \sin \eta + y \cos \eta)}{(x \cos \eta - y \sin \eta)} \right] \\ &= \arctan \left[ \frac{\frac{(x \sin \eta + y \cos \eta)}{x \cos \eta}}{\frac{(x \cos \eta - y \sin \eta)}{x \cos \eta}} \right] \\ &= \arctan \left[ \frac{\frac{y}{x} + \tan \eta}{1 - \frac{y}{x} \tan \eta} \right] \end{aligned} \quad (2.22)$$

$$\begin{aligned} &= \arctan \frac{y}{x} + \arctan [\tan \eta] \\ &= \arctan \frac{y}{x} + \eta \end{aligned} \quad (2.23)$$

Note that (2.20) uses the trigonometric identity  $\sin^2(x) + \cos^2(x) = 1$  and (2.22) uses the identity  $\arctan x + \arctan y = \arctan \left( \frac{x+y}{1-xy} \right)$ , or see [257]. Thus, rotation in Cartesian space results in a shift in the log-polar space:

$$(x \cos \eta - y \sin \eta, x \sin \eta + y \cos \eta) \rightarrow (\mu, \theta + \eta). \quad (2.24)$$

With both uniform scaling and rotation, a similar result can be shown:

$$(\rho(x \cos \eta - y \sin \eta), \rho(x \sin \eta + y \cos \eta)) \rightarrow (\mu + \log \rho, \theta + \eta). \quad (2.25)$$

Applying the Fourier Transform of the LPM of a signal, and (2.25), imply that uniform scaling and rotation in the spatial domain produce phase shifts in the Fourier-Mellin domain. The magnitude of the Fourier-Mellin Transform is a domain invariant to uniform scaling and rotation.

The Fourier-Mellin Transform may be used to construct a transform domain that is invariant to uniform scaling, translation, and rotation, into which a watermark may

be inserted [224, 258]. The conversion from (spatial domain) scaling and rotation to (Fourier-Mellin domain) phase shifts may also be useful for synchronization and template analysis [223, 259, 260], discussed in the next section.

The disadvantage of relying on invariance is that invariance properties are generally applicable for a relatively small set of coordinate transformations. Attacks which apply coordinate transformations in which the signal transform is not invariant will require the detector to have some other means for synchronization. Invariants for some transformations, such as non-uniform scaling, perspective transformations, and warping, may be difficult to find. There are also implementation issues (such as interpolation and signal transform issues) that can complicate invariant domain watermarking [223, 258].

Synchronization-insensitive watermarking without invariant transformation was proposed in [261]. This technique creates a watermark using patches of colored noise such that the watermark detector does not require precise (sample-to-sample) correspondence to successfully detect the watermark. The technique is promising in that some degree of robustness against synchronization attack was observed, however much more development and evaluation is needed. Security issues were noted in [261].

### **2.6.3 Synchronization and templates**

When watermarked signals are subjected to synchronization attack and the watermark is not insensitive or invariant to these attacks, then synchronization is necessary for successful watermark detection. The watermark detector with synchronization is shown in Figure 2.14. The synchronizer examines the input signal and estimates the correspondence between the coordinates of the input signal and that of an embedded watermark. This correspondence information is provided to the watermark signal detector, which attempts to detect the watermark as described in Section 2.1.3. If watermark detection succeeds, then the payload is extracted (if applicable) and the watermark detection process is complete. If watermark detection

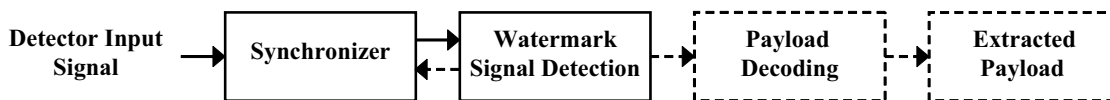


Fig. 2.14. Watermark detection with synchronization

does not succeed, the watermark detector may either “give up” or request another correspondence from the synchronizer and repeat the process. Obviously, the synchronizer is interested in identifying the correspondence with the fewest number of attempts. The synchronization and watermark signal detection processes may be (tightly) coupled and are not necessarily independent.

The synchronization process will not succeed for every input signal. First, the detector’s input is not necessarily watermarked and synchronization should fail for unwatermarked input. Because of the possibility of receiving unwatermarked input, a mechanism or strategy for deciding when the synchronizer “gives up” is necessary. On the other hand, synchronization may (undesirably) fail when a watermarked signal is provided to the detector. Failure to synchronize on watermarked input generally causes the watermark detector to miss. For this reason, the synchronization mechanism itself may be targeted by attacks. Examples of these attacks will be described later.

The watermark signal detector can make use of the information provided by the synchronizer in several ways. The signal detector could detect the watermark directly in the coordinates provided by the synchronizer. The signal detector could also use the synchronizer information to “reverse” any coordinate transformations that may have occurred. For example, the rotation of Figure 2.13(c) may be reversed by rotating the attacked image in the opposite direction. Once the transformations have been reversed, the signal detector detects the watermark in a straightforward manner. More generally, the synchronizer allows the input signal to be transformed to some pre-determined or normalized coordinate system for watermark signal detection.



The most fundamental approach for synchronization is naïve or blind search. In this approach, the watermark detector explicitly searches the space of coordinate transformations to locate the watermark, but without using any special properties of the watermark or other side information. Two examples of the naïve search approach are the exhaustive search and the sliding correlator [3, 242]. The exhaustive search attempts to detect the watermark by searching all possible spatial and temporal coordinate transformations. The sliding correlator attempts to detect the watermark (using correlation) over all spatial and temporal shifts  $x_0, y_0, t_0$ , or

$$\rho = \max_{\forall x_0, y_0, t_0} \sum_x \sum_y \sum_t Z(x - x_0, y - y_0, t - t_0) W(x, y, t). \quad (2.26)$$

The obvious issue with naïve search is that the size or cardinality of the search space is an obstacle for computationally efficient search. The second issue is that naïve search is prone to false positives [262]. Even when the signal detector has a very low false positive rate, the number of attempted detections that occur during a naïve search dramatically increases the likelihood of false positives. False positives make the naïve search approach questionable even when computational cost is not constrained.

The approach generally proposed to address synchronization is the use of *templates*. A template is a pattern which indicates the position of the embedded watermark in the watermarked signal. By examining the template, which is the process known as template matching or template analysis, the synchronizer is able to (quickly) estimate the position of the watermark. Thus, templates enable the synchronizer to use informed search instead of blind search. Several methods for using templates will be described below. In this dissertation, the formal definition of a template is side-information regarding the structure of the watermarked signal which allows the watermark detector to reduce the computational search for synchronization.

Watermarking techniques have proposed three methods for constructing templates. The first method is the embedding of an explicit synchronization signal

into the watermarked signal. The second method is to use the watermark itself as the synchronization signal by placing constraints in the structure of the watermark. The third method is to use salient features of the original signal as the basis for synchronization.

The first method for template construction is the insertion of an auxiliary synchronization signal into the watermarked signal. For example, the synchronization signal may resemble a constellation of peaks [209]. Coordinate transformations that are applied to the watermarked signal affect the synchronization signal in a manner similar to the embedded watermark. Thus, the synchronizer is able to estimate of the position of the watermark by examining where the template resides in the attacked signal. Video watermarking techniques using explicit template embedding include [235, 239, 256], as well as the “helper watermarks” mentioned in [3] and the orthogonal sequences used for temporal synchronization in [263]. Unlike the embedded watermark (which is often noise-like), the synchronization signal is designed to be easily detected or identified by the synchronizer but generally invisible in the spatial domain.

The embedding of the synchronization signal increases the overall distortion of the watermarked signal compared with the original signal. In addition, the synchronization signal is vulnerable to attack. For example, the location of the peaks in a constellation may be estimated and then the peaks removed [264]. Estimation of the template is easier when the same synchronization signal is used for many watermarked signals. Attacks may also insert “fake” synchronization signals or alter an existing signal to lead the synchronizer astray.

The second method for constructing a template is to apply constraints on the structure of the watermark signal to generate the synchronization pattern. An example is a watermark with periodic structure created by repeating or tiling an elementary watermark signal. This structure allows the search for synchronization under spatial shifts to be reduced [38]. In addition, tiled or periodic watermarks have known auto-correlation properties which allow the estimation of rotation and

scale [260, 265, 266]. (Auto-correlation is used for the spatial synchronization technique described in Chapter 4.) The structure of the watermark itself provides the template and no auxiliary synchronization signal is embedded. Unfortunately, the same constraints placed upon the watermark signal to create the template also reduce the capacity and security of the watermark. For example, the redundancy present in a tiled watermark implies that estimating the watermark signal is easier. Other examples of this method are [267–270].

The third method for obtaining a synchronization template is to use salient features of the original signal [212, 236, 271–275] as the basis for synchronization. When the watermark is embedded, a feature detector is used to identify the locations of salient features of the original signal. Example features include the position of edges or corners of an image. The watermark is inserted in accordance to the features. A coordinate transformation applied to the watermarked signal generally affects or moves the features in a similar manner as the watermark. When the watermarked (or attacked) signal is provided to the detector, the synchronizer examines the signal using the feature detector to estimate where the watermark resides.

Synchronization attacks against these templates involve manipulating the features. For example, features may be removed or inserted, or translated (moved). Cropping the watermarked signal may remove features. The manipulation of the watermarked signal must be sufficient to cause a misleading detection by the feature detector (in the synchronizer). Manipulating some features may require some significant alteration to the watermarked signal, but this is dependent on the choice of features used by the watermarking technique.

Some search may be necessary for synchronization despite the use of templates. Template analysis allows the position of an embedded watermark to be estimated but often the degree of precision in the position estimate is limited. Hence, templates reduce but do not generally eliminate the computational search for synchronization.

Temporal synchronization for video watermarking is a relatively unexplored area. Proposed techniques generally embed an “address” or index into the watermarked

video frames [263, 276]. The embedded addresses allow the watermark detector to detect dropped or transposed frames. In [276], only the intra-frames of compressed video are watermarked and the performance was found to be dependent on the degree of motion in the video. In [263], orthogonal sequences are embedded into each video frame which encode the frame's temporal index relative to the video sequence. A correlation based technique is used to recover the index.

This concludes the overview of synchronization. In Chapter 3, temporal synchronization in video watermarking will be explored and models for addressing temporal synchronization will be proposed. Spatial synchronization (for still images) is explored in Chapter 4.

## **2.7 Additional Background**

This section provides a brief overview of some topics of interest. The pseudo-random number generator has been mentioned in the background and their structure and properties are described in Section 2.7.1. State machines are computational devices used in the temporal synchronization models in Chapter 3. A brief overview of state machines appears in Section 2.7.2. Lastly, a few remarks contrasting steganography and watermarking is in Section 2.7.3.

### **2.7.1 Pseudo-random number generators**

The generation of random numbers is useful in many applications, including (Monte Carlo) simulations [277], randomized algorithms [278], recreation (games), cryptography, and watermarking. In a strict sense, random numbers cannot be generated by software alone because the output of a computer program is a deterministic function of its inputs. While the generation of true random numbers lies beyond the reach of software, pseudo-random number generators (PRNGs) produce numbers that “appear” statistically random to applications.

PRNGs have common structure. A PRNG has an internal state and produces output values using the internal state. Each value produced by the PRNG is a function of (only) the state at the instant the value is produced. Furthermore, producing a value also causes the PRNG to change its state. This process is deterministic and lacks any inherent source of “randomness.” The internal state of the PRNG is initialized by the application prior to producing any output values. This process is known as seeding the PRNG and the initial value is known as the *seed*. The consequences of using a PRNG are:

1. The values produced by a PRNG are not random, but are part of a fixed sequence of values.
2. The sequence of values produced by a PRNG is a function of only the seed. Each time the PRNG is seeded with a particular value, then the PRNG will produce the same sequence of output values. Many applications, including watermarking, depend on this “repeatability” property (see below.) Some applications seed the PRNG with a value dependent on a hardware clock or timer, as to cause the PRNG to produce different sequences each time a computer program is executed by the user.
3. An implementation of a PRNG has a finite number of states, which are traversed in a fixed order as the PRNG produces output values. If enough values are produced, then the initial state will be reached again. Thus, the sequence of values produced by a PRNG is periodic. The number of output values produced by the PRNG before the sequence repeats is known as the period of the PRNG. Generally, long period PRNGs are desirable.

There are many methods for pseudo-random number generation, including linear-congruential [103,277,279], feedback shift registers [277], and others [279,280]. These methods may be used to produce pseudo-random sequences that simulate a variety of statistical properties. Some applications desire sequences that appear to be obtained from an independent identically-distributed (i.i.d.) uniform random source.

Other applications desire an i.i.d. Gaussian distributed source, or perhaps sequences with other distributions or correlation properties [232]. There are no guarantees regarding the statistical randomness of the output, which should not be surprising because there is no inherently random process in a PRNG. An application should test the output of a PRNG to verify that the sequence of numbers are suitable for the requirements of the application. A variety of tests [279, 281] have been proposed for evaluating the statistical properties of PRNGs.

In watermarking, PRNGs are often involved in generating the watermark signal. The seed to the PRNG is generally the embedding key  $K_E$ , which places an upper bound on the cardinality of the embedding keyspace as the period of the PRNG. Many techniques directly use output values produced by the PRNG as the watermark signal, while other techniques use the PRNG to perform  $K_E$ -dependent permutations or other types of  $K_E$ -dependent processing. Because the output of a PRNG is dependent only on the seed, the seed value (uniquely) characterizes the pseudo-random sequence, and hence, the watermark. For example, a symmetric watermark detector with the same PRNG used by the watermark embedder is able to recover  $W$  for detection (such as for correlation) with only the parameter  $K_D = K_E$ .

### 2.7.2 State machines

A state machine (SM) is a general computational model for systems that possess memory (or state) [282]. SMs have been used in a variety of applications, including character string recognition and regular-expression matching [278, 283], theoretical computing (such as Turing Machines) [284, 285], and simple “artificial intelligence” engines used in recreational computing [286].

A state machine is defined by the tuple  $(\mathcal{S}, \mathcal{S}_0, \mathcal{I}, \mathcal{O}, \phi, \lambda)$ , where  $\mathcal{S}$  is the set of states,  $\mathcal{S}_0$  is the set of initial states,  $\mathcal{I}$  is the input domain,  $\mathcal{O}$  is the output range,  $\phi$  is the state transition function, and  $\lambda$  is the output function.  $\mathcal{S} = \{s_0, s_1, \dots, s_{|\mathcal{S}|-1}\}$  is the non-empty, countable set of states in the state machine. A “state” is a repre-

sentation of memory, which allows the output and behavior of the SM to depend on current and past inputs. At any given (discrete) time  $t \geq 0$  the state of the machine, known as the current state  $s(t)$ , is exactly one of the members of  $\mathcal{S}$ .<sup>9</sup> The current state of the SM can change in response to the state machine input in accordance to the state transition function. If  $\mathcal{S}$  has finite cardinality, then the SM is known as a finite state machine (FSM).  $\mathcal{S}_0$  is the set of initial states, and is a non-empty subset of  $\mathcal{S}$ . The current state of the SM is initialized to a member of  $\mathcal{S}_0$  when the machine is started, such that  $s(0) \in \mathcal{S}_0$ .  $\mathcal{I}$  is the input domain, which is the set of all possible inputs to the state machine, and  $\mathcal{O}$  is the output range, which is the set of all possible outputs of the state machine. The state transition function  $\phi : \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{S}$  describes the state transitions of the SM. If the current state of the SM is  $s(t) \in \mathcal{S}$ , and the current input is  $i(t) \in \mathcal{I}$ , then the next state will be  $s(t+1) = \phi(s(t), i(t))$ . The output function  $\lambda : \mathcal{S} \rightarrow \mathcal{O}$  is a mapping from each state to an output value. Thus, the output of the state machine is  $\lambda(s(t)) \in \mathcal{O}$ , which is a function of only the current state  $s(t)$ . The output function of some state machines (known as Mealy Machines [282]) are more general, in that  $\lambda$  is also a function of the current input ( $\lambda : \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{O}$ ). However, it is sufficient for the output to depend only on  $s(t)$  for the SMs of interest in this dissertation.

Some state machines are non-deterministic. A non-deterministic state machine may have one or both of the following: multiple starting states, and non-deterministic state transitions. A state machine with multiple starting states has  $|\mathcal{S}_0| > 1$ . When the state machine is initialized before  $t = 0$ , a random member of  $\mathcal{S}_0$  is chosen as the starting state. A state machine with non-deterministic state transitions has a state transition function  $\phi(s, i)$  which returns a non-empty set of possible next states. When a state transition occurs, the state machine transitions to a randomly chosen member of that set.

---

<sup>9</sup>Note the difference between the notation  $s_i$  and  $s(t)$ .  $s_i$  refers to a specific member of  $\mathcal{S}$  (the  $i$ -th state) and is independent of time.  $s(t)$  is the current state of the state machine at time  $t$ .

A state machine is often illustrated or represented using a directed graph, where the vertices corresponds to the states and the edges correspond to the state transition function  $\phi(\cdot)$ .

### 2.7.3 Steganography

Watermarking and steganography are often mentioned together, and indeed, robust watermarking techniques may be used in some steganographic applications. In this section, these related areas are (briefly) contrasted.

The objective of steganography [47] is to create a covert channel. Steganographic techniques create the channel by embedding information into innocuous cover signals. The signal containing the embedded information, known as the stego-signal, appears identical to the cover signal and may be transmitted through an insecure channel or even publicly distributed. However, only parties that are aware of the hidden communications are able to (easily) recover the embedded message from a stego-signal. There may be censors, sometimes known as wardens, who examine channels for the purpose of identifying possible stego-signals. Distinguishing stego-signals from signals that contain no hidden information is a part of steganalysis [287].

The parallels between steganography and watermarking are straightforward. The hidden message is the payload, the cover signal is the original signal, and the stego-signal is the watermarked signal. Embedding hidden information is a process similar to watermark embedding, and extracting hidden information from a stego-signal is a process similar to watermark detection.

The difference between watermarking and steganography lies in the purpose and requirements of the embedded watermark. For steganography, the most important objective is to avoid suspicion of the stego-signal. Therefore, the degree of invisibility required for information embedding is generally much greater for steganography than robust watermarking. This arises under the assumption that wardens are likely to use statistical analysis [287–290] to identify possible stego-signals, and thus infor-



mation embedding for steganography must not only be perceptually invisible, but also *statistically invisible*. Robustness may be a concern in some steganography applications, but generally perceptual and statistical invisibility is of much greater importance than robustness. On the other hand, robust watermarking applications are generally more concerned that an embedded watermark is difficult to remove. Perceptual transparency is important, but generally not to the degree in which users cannot differentiate watermarked and unwatermarked signals with statistical analysis. In fact, users may be quite aware that signals are watermarked. For example, a video distributor may warn users that all videos are watermarked to protect the interests of the copyright holder. Another difference is that in robust watermarking applications, the embedded watermark is generally related to the watermarked signal in some way, such as encoding the source or intended recipient of the watermarked signal. In steganography, it is not unusual for the embedded message to be completely unrelated to the cover signal.

### 3. TEMPORAL SYNCHRONIZATION

In this chapter, a framework is developed for temporal synchronization in blind symmetric video watermarking [291]. New models are proposed for watermark embedding and detection that apply to a large class of video watermarking techniques. The models demonstrate that temporal synchronization is challenging for video watermarks lacking temporal redundancy. Efficient temporal synchronization is achievable by designing watermarks with temporal redundancy and allowing a limited search by the watermark detector. Experimental results obtained from an implementation of the models show agreement with the theoretical foundations. Spatial synchronization issues are not considered in this chapter.

#### 3.1 Temporal Synchronization Framework

In this section, the foundation for exploring temporal synchronization is developed from the classical model of watermarking. The classical model was described in Section 2.1 (see also Figure 2.1) and is adapted here for video watermarking, using notation that indicates the temporal structure of video more explicitly. Under the classical model, a watermark is inserted into an original video to produce the watermarked video. The watermarked video may be subjected to attack. Finally, the watermarked and possibly attacked video is provided to the watermark detector.

The watermark embedder is provided three inputs: The original video  $\mathbf{X}$ , the embedding key  $K_E$ , and payload  $M$ . The original video is an ordered sequence  $\mathbf{X} = \langle X(0), X(1), X(2), \dots, X(t), \dots \rangle$ , with  $X(t)$  corresponding to the (two-dimensional) frame displayed at time  $t$ . For convenience,  $t$  is discrete and expressed in units of frames, so  $t$  can also be referred to as the frame index. The first frame of the video is indexed by  $t = 0$ . The embedding key  $K_E$  and message  $M$  are identi-

cal to Section 2.1.1. The output of the embedder is the watermarked video  $\mathbf{Y} = \langle Y(0), Y(1), \dots, Y(t), \dots \rangle$ .

The watermarked video may be attacked. The primary focus here is temporal synchronization attacks, which were introduced in Section 2.6.1. These attacks alter the temporal structure of the watermarked video and may confuse the watermark detector. Let  $\hat{\mathbf{Y}} = \langle \hat{Y}(0), \hat{Y}(1), \dots, \hat{Y}(t), \dots \rangle$  denote the watermarked and possibly attacked video that is provided to the detector. If the video has not been attacked, then  $\hat{\mathbf{Y}}$  is identical to  $\mathbf{Y}$  and  $\hat{Y}(t) = Y(t)$  for all  $t$ . If the video is attacked then it is not necessarily true that  $\hat{Y}(t) = Y(t)$  for any  $t$ . For example, frame dropping attacks remove frames from  $\mathbf{Y}$ , while a frame insertion attacks insert arbitrary frames into  $\mathbf{Y}$  to produce  $\hat{\mathbf{Y}}$ .

The watermark detector examines its input video  $\mathbf{Z}$  and determines if the watermark is present. While any signal could be provided to the detector, the interest here is when  $\mathbf{Z} = \hat{\mathbf{Y}}$ . The following assumptions are made with respect to watermark detection:

1. Watermark detection is blind and the original signal  $\mathbf{X}$  is not available to the detector. Blind detection is more general than non-blind detection because the detector has less information for watermark detection.
2. The watermarking technique is symmetric, with  $K_D = K_E$ . Thus, the keyspace can be written as  $\mathcal{K} = \mathcal{K}_E = \mathcal{K}_D$ . Like the classical model, the cardinality of the keyspace is assumed to be very large. More specifically,  $\mathcal{K}$  is assumed sufficiently large that a computational search through  $\mathcal{K}$  is not feasible.
3. The watermark detector does not have access to the entire input video, but only a relatively small portion of the video by which to detect the watermark. As mentioned in Section 2.5.2, this is a common condition in video watermarking and the complications in synchronization described in Section 2.6 are applicable. Thus, the watermark detector performs initial synchronization, and then

continues to detect the watermark as the input signal is available, or until the watermark detector loses synchronization.

While the classical model is useful for an overview of watermarking, insight in the temporal synchronization problem requires a more detailed examination of the temporal structure of video watermarks. The discussion will now extend from the classical model, leading to new models for watermark embedding and detection.

When a watermark is embedded in a video sequence, one of the parameters that determines its structure is the embedding key  $K_E$ . In video watermarking,  $K_E$  is generally used to create a *key schedule* or *key sequence*  $\mathbf{K} = \langle K(0), K(1), \dots, K(t), \dots \rangle$ , which is the ordered sequence of sub-keys for generating the watermark embedded in the individual frames of  $\mathbf{Y}$ . Specifically,  $K(t)$  is the sub-key used for generating the watermark signal  $W(t)$  embedded in frame  $Y(t)$ . Many video watermarking techniques use  $K_E$  as the seed of a PRNG (see Section 2.7.1), which produces the watermark signal  $\mathbf{W}$  directly or performs  $K_E$ -dependent processing that produces the watermarked signal. For these techniques,  $K(t)$  corresponds to the internal state of the PRNG when frame  $X(t)$  is watermarked. Other watermarking techniques explicitly embed the same watermark signal in each frame of video, and for these techniques  $K(t) = K_E$  for all  $t$ . This assumption, where a key sequence is produced from  $K_E$ , holds for a large class of video watermarking techniques, including [38, 61, 89, 104, 212, 233–235, 242, 243].  $K_E$  and  $K(t)$  are assumed to be members of the key space  $\mathcal{K}$ .

Let the ordered sequence  $\hat{\mathbf{K}} = \langle \hat{K}(0), \hat{K}(1), \dots, \hat{K}(t), \dots \rangle$  denote the keys used to produce the watermark signals  $\hat{\mathbf{W}} = \langle \hat{W}(0), \hat{W}(1), \dots, \hat{W}(t), \dots \rangle$  embedded in the frames of  $\hat{\mathbf{Y}}$ , where  $\hat{K}(t)$  is the key used to generate the watermark  $\hat{W}(t)$  embedded in frame  $\hat{Y}(t)$ . If the watermarked video has not been attacked, then  $\hat{Y}(t) = Y(t)$  and thus  $\hat{K}(t) = K(t)$  for all  $t$ . However, temporal synchronization attacks such as frame dropping, insertion, and transposition alter the sequence of frames in the video, causing changes to  $\hat{K}(t)$ . Signal processing attacks that do not affect the temporal structure of the watermarked signal, such as spatial filtering, affect  $\hat{\mathbf{Y}}$  but not  $\hat{\mathbf{K}}$ .

$\hat{K}(t)$  and  $\hat{W}(t)$  are not defined if  $\hat{Y}(t)$  is an arbitrary, unwatermarked frame that has been inserted into the video (as an attack.)

The objective for the watermark detector is to determine  $\hat{K}(t)$  when frame  $\hat{Y}(t)$  is examined. (A blind detector does not have access to  $Y(t)$ .) The process of finding  $\hat{K}(t)$  when frame  $\hat{Y}(t)$  is examined is temporal synchronization. If the detector can determine  $\hat{K}(t)$ , then temporal synchronization is achieved. If the detector cannot determine  $\hat{K}(t)$ , then temporal synchronization is lost. Note that the watermark detector can still miss (such as by interference) when temporal synchronization is successful.

If an attacker can determine  $\hat{K}(t)$  or  $\hat{W}(t)$  for any frame  $\hat{Y}(t)$ , then the structure of the watermark has been deduced and watermark security is broken. As discussed in Section 2.2.4, watermark estimation permits unauthorized removal and may permit unauthorized embedding. One way to examine security is to consider how difficult the watermark is to estimate. The watermark detector does have an advantage compared to an attacker, in that the detector is provided with  $K_D = K_E$ .

The key schedule can have a dramatic effect on the ease of synchronization and the security of the watermark. To illustrate the effects of the key schedule on synchronization and security, three classes of watermarks with special key sequences are discussed: The time invariant key, time independent key, and periodic key watermarks.

A time invariant key watermark uses the same key to construct and embed the watermark into each frame of the video. Temporal synchronization for this class of watermarks is a simple matter because the detection key for any watermarked frame of the video is known to be  $K_E = \hat{K}(t) = K(t)$  for all  $t$  without performing search. However, time invariant key watermarks may be vulnerable to estimation by temporal averaging, and an attacker that successfully obtains the watermark for a single frame of the video breaks the security of the watermark.

For a time independent key watermark, the keys used to construct and embed the watermark signal in successive video frames are nearly independent. Such a

key schedule may be produced by using a PRNG to create the watermark signal for each frame, whose internal state is never re-seeded after the initial seeding with  $K_E$ . Strictly speaking, the watermark keys  $K(t_1)$  and  $K(t_2)$ ,  $t_1 \neq t_2$ , are not truly independent because a single key  $K_E$  is used to produce all the watermarking keys in the key schedule. The keys in the time independent key schedule, however, do not repeat or repeat with an extremely long period. Knowledge of the key used to embed the watermark in one frame yields little information about the key used to embed the watermark in other frames, however, the detector must generally search  $\mathcal{K}$  to find  $\hat{K}(t)$  when synchronization is lost. Watermark robustness against temporal collusion attacks may also be an issue (see Section 2.5.2.)

In a time periodic key watermark, the keys of the key schedule form a repeating sequence (with relatively short period). In such a key schedule, every  $K(t)$  (and thus every  $\hat{K}(t)$ ) is a member of a small set of keys  $\mathcal{K}' \subset \mathcal{K}$  (with  $|\mathcal{K}'| \ll |\mathcal{K}|$ ) that are repeated. A search over  $\mathcal{K}'$  may not be infeasible. With the repeated keys, estimating the watermark requires more effort than time-invariant key watermarks but much less effort than for time-independent key watermarks. For example, estimating the period of the watermark key sequence may be possible by (auto-)correlation, and vector quantization or clustering techniques [82] may be used to estimate the embedded watermark from  $\hat{\mathbf{Y}}$ . Once an attacker obtains the key sequence over a single period, the attacker has obtained the entire key sequence of the watermark.

The time invariant key, independent key, and periodic key sequences differ in the amount of *temporal redundancy* in the key schedule. Temporal redundancy in the key schedule refers to the degree of “randomness” of the keys in the key schedule. The sequence of keys appearing in a key schedule with low temporal redundancy (such as the time-independent key schedule) is more random and recovering synchronization may require a search over  $\mathcal{K}$ . A key schedule with greater temporal redundancy allows the detector to reduce the search needed for synchronization. For example, the detector may be able to predict likely values for an unknown  $\hat{K}(t)$ , based on the past keys  $\hat{K}(t-1), \hat{K}(t-2), \dots$ . However, there is a security trade-off in that greater

temporal redundancy in the key schedule can make deducing  $\hat{K}(t)$  or estimating the embedded watermark easier for an attacker. Temporal redundancy shall be revisited in the design of key schedules that are resistant to frame dropping and transposition attacks in Section 3.5. In the next sections, models for watermarking are introduced which encompasses the time invariant key, time independent key, and periodic key watermarks as special cases.

### 3.2 Watermark Embedding Model

To model the construction of the key schedule, the classical watermark embedder is extended as shown in Figure 3.1. The function of the watermark signal generator and watermark insertion are identical to that of watermark generation and insertion, respectively, in the classical model. The embedder model assumes that the watermark is generated on a frame-by-frame basis, but does not depend on the structure or design of the watermarks embedded in each video frame. Any watermark construction and embedding techniques may be used to produce the embedded watermark for each frame of the video. The key generator and the feature extraction are new components, and will be described in detail below.

The overall steps for watermark embedding are as follows. For each frame of the video:

#### Watermark Embedding Procedure (WEP)

1. The key generator provides  $K(t)$ , the key used to watermark the current frame, to the watermark signal generator.
2. The watermark signal generator uses  $K(t)$  to produce  $W(t)$ , the watermark signal to be embedded into the current frame. For example,  $K(t)$  may be used to seed a pseudo-random number generator which produces  $W(t)$ . Like the classical model, the watermark signal may be dependent on the payload  $M$  as well as the current video frame  $X(t)$ .

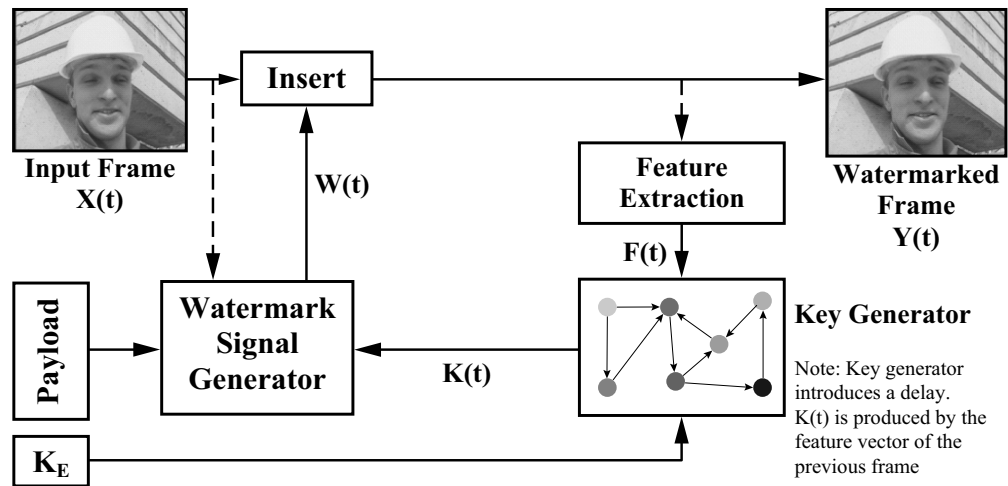


Fig. 3.1. Watermark embedding model



3. The watermark embedder inserts  $W(t)$  into the original video frame  $X(t)$  to produce the watermarked video frame  $Y(t)$ .
4. The feature extractor examines the watermarked video frame and produces a feature vector  $F(t)$ .<sup>1</sup>
5. The key generator produces the watermark key for the next frame,  $K(t + 1)$ . To produce  $K(t + 1)$ , the key generator uses  $K_E$  and  $F(t)$ . Return to step 1 for the next frame of the video.

The purpose of the key generator is to produce the watermarking key for each frame of the video, and thus, produce the key schedule for the watermark. The key generator is modeled using a state machine (SM), introduced in Section 2.7.2. The state machine may be non-deterministic.

When an SM is used as a key generator, the SM accepts fixed  $K_E$  and frame-dependent feature vector  $F(t)$  as inputs and produces a key value as output. Thus, the input domain  $\mathcal{I}$  will generally be the Cartesian product of the key space  $\mathcal{K}$  and the range of all possible feature vectors from the feature extractor. The output range is the key space  $\mathcal{O} = \mathcal{K}$ . The starting state may depend on  $K_E$ . In step 1 of the WEP, the state machine outputs  $\lambda(s(t))$ . State transitions occur in step 5, where  $s(t + 1) = \phi(s(t), K_E, F(t))$ . After the state transition, the key  $K(t + 1)$  is obtained by  $\lambda(s(t + 1))$ . The output mapping function  $\lambda(\cdot)$  of a key generator must also be one-to-one.

The feature extractor examines each watermarked video frame  $Y(t)$  and produces a feature vector  $F(t)$ . The feature vector is then provided to the key generator, which allows the key sequence to be video-dependent. A video-dependent key schedule can increase the difficulty of inverting the watermark and provide some benefit against ownership and copy attacks (see Section 2.1.2.) However, the disadvantage of using features is that the dependence of the key schedule on the features imply that

---

<sup>1</sup>Strictly speaking,  $F(t)$  is a vector. However, it is a vector obtained from observing a single frame,  $Y(t)$ .  $F(t)$  is not written as  $\mathbf{F}(t)$  to avoid confusion with  $\mathbf{X}$ ,  $\mathbf{Y}$ ,  $\hat{\mathbf{Y}}$ , and other symbols whose elements form an ordered sequence across time (i.e. over multiple frames of video).

temporal synchronization can be lost by performing attacks that change the feature vectors. For robust video watermarking, the features should be chosen such that  $F(t)$  changes in value only when significant alterations are made to the watermarked frame  $Y(t)$ . Ideally, the features should be sufficiently robust so that such an attack is successful only when the attacked video no longer has any value in the application. For security, the features should be dependent on an auxiliary key. The features should also be computationally efficient to obtain. Feature extraction may be omitted if the key generator does not use  $F(t)$ .

The modeling of the key generator using a state machine provides generality for the embedder model. While video watermarking techniques generally do not (explicitly) mention the use of a state machine, most (and arguably, nearly all) techniques use a PRNG for watermark generation. A PRNG is a state machine that produces a (nearly) time-independent key sequence. Examples of the time-invariant key, time-independent key, and time-periodic key generators are shown to illustrate these key sequences as special cases of the model. These examples are not unique and there are other key generator configurations which can produce these key schedules. None of the watermark embedders for these key schedules use feature extraction. Assume the key space is  $\mathcal{K} = \{0, 1, 2, \dots, |\mathcal{K}| - 1\}$  and the embedding key is  $K_E \in \mathcal{K}$ . With the exception of the state transition function, the state machines for these examples are common:  $\mathcal{S} = \{s_0, s_1, \dots, s_{|\mathcal{K}|-1}\}$ ,  $\mathcal{S}_0 = \{s_{K_E}\}$ ,  $\mathcal{I} = \mathcal{K}$ ,  $\mathcal{O} = \mathcal{K}$ , and  $\lambda(s_i) = i$ .

The key generator for a time-invariant key watermark has the state transition function

$$\phi(s_i, K_E) = s_i \tag{3.1}$$

For a time-independent key watermark, the key generator has state transition function

$$\phi(s_i, K_E) = \begin{cases} s_{i+1}, & s_i \in \{s_0, s_1, \dots, s_{|\mathcal{K}|-2}\} \\ s_0, & s_i \in \{s_{|\mathcal{K}|-1}\} \end{cases} \tag{3.2}$$

For a periodic key watermark with period  $T > 1$  frames, the key generator has state transition function

$$\phi(s_i, K_E) = \begin{cases} s_{i+1}, & s_i \in \{s_{K_E}, s_{K_E+1}, \dots, s_{K_E+T-2}\} \\ s_{K_E}, & \text{otherwise} \end{cases} \quad (3.3)$$

for  $K_E \in \{0, 1, \dots, |\mathcal{K}| - T\}$ .<sup>2</sup> The graph representation of these state machines is shown in Figure 3.2.

The computational cost of watermark embedding is expressed by

$$\Phi_{\text{embed}} = \Phi_{\text{generate}} + \Phi_{\text{insert}} + \Phi_{\text{feature}} \quad (3.4)$$

where  $\Phi_{\text{embed}}$  is the embedding cost per video frame,  $\Phi_{\text{generate}}$  is the cost for generating the watermark signal,  $\Phi_{\text{insert}}$  is the cost for watermark insertion, and  $\Phi_{\text{feature}}$  is the cost for feature extraction.  $\Phi_{\text{generate}}$  and  $\Phi_{\text{feature}}$  generally depend on the size of the video frame while  $\Phi_{\text{insert}}$  depends on the (choice of) features. The cost of key generation (the state machine) is neglected in (3.4), as this cost is generally negligible.

### 3.3 Watermark Detection Model

This section describes a model for symmetric blind video watermark detection. The detector model is shown in Figure 3.3, under the assumption that the test signal is the attacked video  $\mathbf{Z} = \hat{\mathbf{Y}}$ . Symmetric watermark detection also has the detection key  $K_D = K_E$ . The major components of the detector model are the watermark signal detector, feature extractor, state predictor, detector control, and the queue. The detector has knowledge of the embedder's key generator  $(\mathcal{S}, \mathcal{S}_0, \mathcal{I}, \mathcal{O}, \phi, \lambda)$  and feature extraction.

The watermark signal detector is identical to that of the classical model. It is the watermark detector corresponding to the technique used to embed the watermark signal. The watermark signal detector is provided  $\hat{Y}(t)$  and a detection key, and

<sup>2</sup>The state transition function for  $K_E \in \{|\mathcal{K}| - T + 1, \dots, |\mathcal{K}| - 1\}$  is conceptually similar to (3.3), but requires the indices of the states in the period to “wrap-around”: There is no state  $s_{|\mathcal{K}|}$  in  $\mathcal{S}$ , so the next state after  $s_{|\mathcal{K}|-1}$  is  $s_0$ .

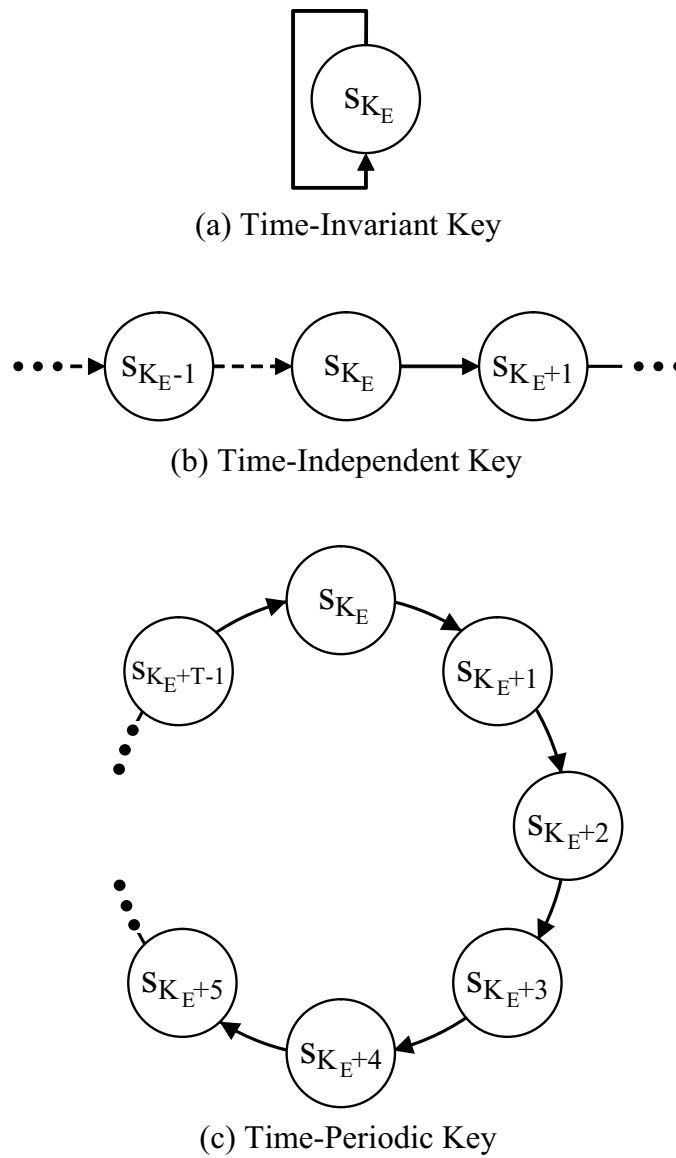


Fig. 3.2. Example key generators

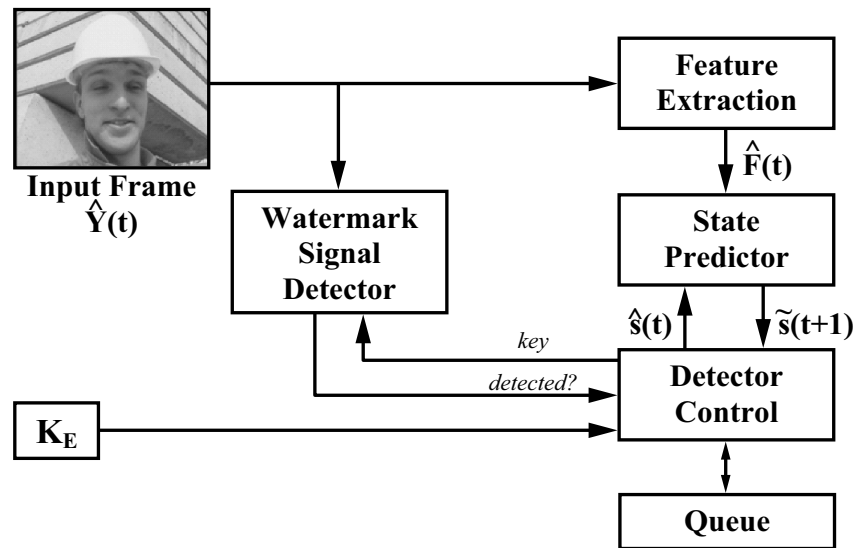


Fig. 3.3. Watermark detection model

determines whether or not the watermark is present in the frame using the key. The detection key is obtained from the detector control. The watermark signal detector may be invoked multiple times for a single frame of video.

The feature extraction is identical to that of the embedder. The detector’s feature extractor examines the input frame  $\hat{Y}(t)$  and produces a feature vector  $\hat{F}(t)$ , which is provided to the state predictor.

The state predictor and the queue are the significant components for watermark detection and synchronization. At this time, the structure of the the state predictor and the queue are described. The underlying synchronization mechanism will be discussed later. The state predictor accepts a state input  $\hat{s}(t)$ , the embedding key  $K_E$ , and feature input  $\hat{F}(t)$  and outputs  $\tilde{s}(t + 1)$ , the predicted state for  $\hat{s}(t + 1)$ . The prediction function is identical to the state transition function of the watermark embedder:  $\tilde{s}(t + 1) = \phi(\hat{s}(t), K_E, \hat{F}(t))$ . The state predictor is not a state machine.

The queue serves as the detector’s memory. A queue is an array of elements  $Q = \langle q(0), q(1), \dots, q(|Q| - 1) \rangle$ , where each element may contain some data. The queue size  $|Q|$  is number of elements present in the queue and the queue capacity is the maximum size of the queue. The element  $q(0)$  is known as the head, and  $q(|Q| - 1)$  is known as the tail. When new data is inserted into the queue, all of the data in the queue shift one element and the new data is placed at the head of the queue. If the queue was full prior to the insertion, then any data that resided at the tail of the queue is lost. This type of queuing behavior is known as a FIFO (First-In-First-Out) queue. In addition to the insertion of new data, the queue supports moving or “promoting” any data present in the queue to the head of the queue. When data in a specific queue element is promoted, all the other data in the queue are moved—preserving their relative order—to make room at the head of the queue and the promoted data is placed into the head. For the watermark detector queue, the data stored in the queue will be members of  $\mathcal{S}$ . Prior to receiving the first frame of video, the watermark detector queue is empty.

The detector control component manages the operation of all the components of the watermark detector, performing the steps of the Watermark Detection Procedure (WDP).

### Watermark Detection Procedure (WDP)

1. The input frame  $\hat{Y}(t)$  is provided to the watermark signal detector and the feature extractor. The feature extractor examines  $\hat{Y}(t)$  and obtains  $\hat{F}(t)$ .
2. For each state  $s_{init} \in \mathcal{S}_0$ , the detector sends the key  $\lambda(s_{init})$  to the watermark signal detector. If the detector finds the watermark using key  $\lambda(s_{init})$ , let  $\hat{s}(t) = s_{init}$  and continue to step 5. If the detector fails to find the watermark using the keys of all states in  $\mathcal{S}_0$ , continue to the next step.
3. For each state  $s_q \in Q$ , the detector sends the key  $\lambda(s_q)$  to the watermark signal detector. If the detector finds the watermark using  $s_q$ , let  $\hat{s}(t) = s_q$ , promote  $s_q$  to the head of the queue, and continue to step 5. If the detector fails to find the watermark using the keys of all states in the queue, continue to the next step.
4. At this step, either  $\hat{Y}(t)$  is not watermarked or temporal synchronization failed for the current frame. Return to step 1 and try synchronizing with the next frame,  $\hat{Y}(t+1)$ .
5. At this step,  $\hat{K}(t)$  was found and temporal synchronization is achieved for the current frame.  $\hat{s}(t)$  is the state which maps to  $\hat{K}(t)$  under  $\lambda(\cdot)$ . Use the state predictor to predict the state  $\tilde{s}(t+1) = \phi(\hat{s}(t), K_E, \hat{F}(t))$ . If the predicted  $\tilde{s}(t+1)$  is already in the queue, promote it to the head. Otherwise insert the predicted  $\tilde{s}(t+1)$  into the queue. Then return to step 1 for the next frame of video. If  $\phi(\cdot)$  returns a set of states (a non-deterministic SM), then this step is repeated for every member of  $\phi(\hat{s}(t), K_E, \hat{F}(t))$ , before returning to step 1.

A glance at the WDP shows that the queue is used to perform a limited search to find the appropriate detection key for each frame of the video. The queue stores the

states which produced recently detected keys, and their next states. The next states are obtained by using the state predictor. Watermark detection only involves a single frame at a time, which fulfills the assumed condition that the entire test signal is not available to the detector. A more detailed analysis of the synchronization mechanism is presented in Section 3.4.

The watermark detection model generalizes the behavior of video watermark detectors, analogous to the watermark embedding model described in Section 3.2 generalizing a symmetric blind video watermark embedder. Video watermark detectors generally perform the steps of the WDP in principle, even when the watermark detection technique is not described with explicit mention of the watermark detection model. In particular, the watermark detector possesses side-information regarding the watermark embedding process. This side-information generally allows the detector to generate the watermark signal (step 3 of the WDP) and then detect for the presence of the watermark signal in the input video frame. If the detector discovers that  $\hat{K}(t)$  was the correct key to watermark frame  $\hat{Y}(t)$ , the detector assumes (i.e. predicts)  $\hat{K}(t+1)$  by using this side-information (step 5 of the WDP) and searches for this key in subsequent frames of the video. Watermark detection continues in such a manner until the detector becomes confused by a synchronization attack. When the detector loses synchronization, and for initial synchronization, the detector searches for a particular key to resynchronize (step 2 of the WDP).

The computational cost for watermark detection (per frame) is given by

$$\Phi_{\text{detect}} = \kappa [\Phi_{\text{signal-detect}}] + \Phi_{\text{feature}}, \quad (3.5)$$

where  $\kappa$  is the sum of the queue capacity and the cardinality of the set  $\mathcal{S}_0$ , and  $\Phi_{\text{signal-detect}}$  is the computational cost for watermark signal detection.  $\Phi_{\text{signal-detect}}$  is generally dependent on the size of a video frame. The detector control and state prediction generally have negligible cost and have not been included in (3.5). Clearly, the detection cost is linear with respect to the queue capacity, which represents the cost of performing the limited search for synchronization. If computational resources are available, the individual signal detections may be implemented in parallel.



### 3.4 Analysis

The function of the embedder and detector models are examined in this section. It is shown that the detector will detect the watermark embedded in every frame of the watermarked video without attacks and that the insertion of arbitrary (un-watermarked) frames will not desynchronize the detector. The vulnerability of the watermark detector to synchronization loss under frame dropping or transposition attack is also shown.

The objective of the detector is to determine  $\hat{K}(t)$  when frame  $\hat{Y}(t)$  is examined. However, for a watermark produced by the watermark embedder discussed in Section 3.2,  $\hat{K}(t) = \lambda(\hat{s}(t))$ , where  $\hat{s}(t)$  is the underlying state of the key generator which produced  $\hat{K}(t)$ . The function  $\lambda(\cdot)$  is one-to-one, and is only a function of the state. Thus, the objective of the detector can be restated as to determine  $\hat{s}(t)$  when frame  $\hat{Y}(t)$  is examined. The analysis will focus on the states, specifically the states of the embedder's key generator to produce the watermark and the states in the detector's queue.

The watermark signal detector is assumed to be always correct for any key and input video frame (no false positives and no misses). Also, the analysis shall assume that the key generator uses a deterministic state machine. This is to simplify the presentation; a similar analysis can be performed for non-deterministic state machines, which would be complicated by  $\phi(\cdot)$  returning a set of states as opposed to a single state for a deterministic SM, but the general principle remains the same.

Some lemmas about the properties of the embedder and detector models:

- **Lemma 1 (Fundamental Watermark Structure)** *Let  $Y(t)$  and  $Y(t + 1)$  be any two successive frames of the watermarked video. Assume that  $K(t) = \lambda(s(t))$  produced the watermark signal embedded in  $Y(t)$  and  $K(t + 1) = \lambda(s(t + 1))$  produced the watermark signal embedded in  $Y(t + 1)$ . Assume that  $F(t)$  is obtained from the feature extractor from  $Y(t)$ , and  $K_E$  is constant. Then,  $s(t + 1) = \phi(s(t), K_E, F(t))$ .*

**Proof** This follows from the WEP. In particular, exactly one state transition occurs for every pair of successive frames, and for any state transition  $s(t+1) = \phi(s(t), K_E, F(t))$ . ■

- **Lemma 2** *Suppose  $\hat{Y}(t)$  and  $\hat{Y}(t+1)$  are two successive watermarked frames of the detector's input video and neither frame is attacked or involved in an attack. Assume that  $\hat{K}(t) = \lambda(\hat{s}(t))$  produced the watermark signal embedded in  $\hat{Y}(t)$  and  $\hat{K}(t+1) = \lambda(\hat{s}(t+1))$  produced the watermark signal embedded in  $\hat{Y}(t+1)$ . Assume that  $\hat{F}(t)$  is obtained from the detector's feature extractor at time  $t$ , and  $K_E$  is constant. Then,  $\hat{s}(t+1) = \phi(\hat{s}(t), K_E, \hat{F}(t))$ .*

**Proof** Because neither  $\hat{Y}(t)$  nor  $\hat{Y}(t+1)$  is attacked, there must exist a pair of frames  $Y(u)$  and  $Y(u+1)$  such that  $\hat{Y}(t) = Y(u)$  and  $\hat{Y}(t+1) = Y(u+1)$ . (The time indices  $t$  and  $u$  are not necessarily identical because frames may have been dropped or inserted into  $\hat{Y}$  outside the time interval  $t$  and  $t+1$ .) From  $\hat{Y}(t) = Y(u)$ , it follows that  $\hat{K}(t) = K(u)$  and  $\hat{s}(t) = s(u)$ . The feature extraction will also produce identical feature vectors  $\hat{F}(t) = F(u)$ . From  $\hat{Y}(t+1) = Y(u+1)$ , it follows that  $\hat{K}(t+1) = K(u+1)$  and  $\hat{s}(t+1) = s(u+1)$ . Lemma 1 holds for  $Y(u)$  and  $Y(u+1)$ , so  $s(u+1) = \phi(s(u), K_E, F(u))$ . Then,  $\hat{s}(t+1) = s(u+1) = \phi(s(u), K_E, F(u)) = \phi(\hat{s}(t), K_E, \hat{F}(t))$ . ■

- **Lemma 3 (Detector Behavior)** *Let  $\hat{Y}(t)$  be the frame of the video examined by the detector. Then, temporal synchronization is achieved if and only if  $\hat{Y}(t)$  is watermarked, and  $\hat{s}(t)$  is either a member of  $\mathcal{S}_0$  or in the queue at time  $t$ , where  $\hat{s}(t)$  is the state which produced  $\hat{K}(t) = \lambda(\hat{s}(t))$ .*

**Proof** Suppose temporal synchronization is achieved. This implies that the detector found some key  $\hat{K}(t)$  and state  $\hat{s}(t)$  such that  $\hat{K}(t) = \lambda(\hat{s}(t))$  produced the watermark signal embedded in frame  $\hat{Y}(t)$ . To establish synchronization, the WDP searches the keys corresponding to the states in  $\mathcal{S}_0$  (during step 2)

and the queue (during step 3), and no other states. Thus, it must be the case that  $\hat{s}(t)$  is either in  $\mathcal{S}_0$  or the queue. Suppose that  $\hat{Y}(t)$  is watermarked with key  $\hat{K}(t) = \lambda(\hat{s}(t))$ , and  $\hat{s}(t)$  is either a member of  $\mathcal{S}_0$  or is in the queue at time  $t$ . Because the WDP attempts to detect the watermark using all of the states in  $\mathcal{S}_0$  and in the queue, the watermark detector will try  $\hat{s}(t)$  and temporal synchronization is achieved. ■

- **Lemma 4 (Initial Synchronization)** *Let  $\hat{Y}(t)$  be the frame of the video examined by the detector, watermarked by  $\hat{K}(t) = \lambda(\hat{s}(t))$  and  $\hat{s}(t) \in \mathcal{S}_0$ . Temporal synchronization will be achieved.*

**Proof** This follows immediately from Lemma 3. The significant statement of this Lemma is that synchronization will be achieved regardless of the contents of the queue. ■

- **Lemma 5 (Queue Lemma I)** *Let  $\hat{Y}(t)$  be the watermarked frame examined by the detector, where  $\hat{K}(t) = \lambda(\hat{s}(t))$  produced the watermark embedded in the frame. Suppose temporal synchronization is achieved. Then, for frame  $\hat{Y}(t+1)$ , the state at  $q(0)$  (the head of the queue) is  $\phi(\hat{s}(t), K_E, \hat{F}(t))$ , and if  $\hat{s}(t) \notin \mathcal{S}_0$  and the queue is sufficiently large, the state at  $q(1)$  is  $\hat{s}(t)$ .*

**Proof** By lemma 3, if temporal synchronization is achieved then  $\hat{s}(t)$  must either be in  $\mathcal{S}_0$  or the queue. The remainder of the proof follows by inspecting the queue after step 5 of the WDP. ■

When the key generator uses a non-deterministic state machine, this lemma should be restated such that when  $\hat{s}(t) \notin \mathcal{S}_0$  and the queue is sufficiently large, then the state  $\hat{s}(t)$  will be in the queue at  $q(|\phi(\hat{s}(t), K_E, \hat{F}(t))| + 1)$ . A sufficiently large queue capacity implies that the queue has enough elements to store the current state and all next states. This is a queue capacity of 2 for

a deterministic SM key generator, but larger for a SM with non-deterministic state transitions. Specifically, the queue capacity must be at least the maximum number of next states returned by  $\phi(\cdot)$  for any  $s(t)$ ,  $K_E$ , and  $F(t)$ , plus one additional element (for the current state  $s(t)$ .)

- **Lemma 6 (Queue Lemma II)** *Let  $\hat{Y}(t)$  be the frame of the video examined by the detector. Suppose temporal synchronization is not achieved. Then the queue is unchanged.*

**Proof** If temporal synchronization fails, then either  $\hat{Y}(t)$  is not watermarked or the state that produces the key which generated the watermark signal embedded in  $\hat{Y}(t)$  is not in  $\mathcal{S}_0$  or the queue. In this case, none of the steps in the WDP affect the queue. ■

**Theorem 3.4.1 (Correctness of watermark detector with no attacks)** *Let  $\hat{Y}$  be the watermarked video provided to the detector. Suppose no attacks are performed on  $\hat{Y}$ . Then the watermark detector will achieve synchronization and detect the watermark in every frame of the video.*

**Proof** The proof will be by induction on  $t$ . Base Case:  $\hat{Y}(0)$  is the first frame of the video. There are no attacks, so  $\hat{Y}(0) = Y(0)$ , which was watermarked by the key generated by state  $s(0) \in \mathcal{S}_0$ . By Lemma 4, temporal synchronization will be achieved.

Inductive Case: Suppose temporal synchronization is achieved for frame  $\hat{Y}(t)$ . By Lemma 5,  $\phi(\hat{s}(t), K_E, \hat{F}(t))$  will be at the head of the queue. By Lemma 2, the state for the next frame is  $\hat{s}(t+1) = \phi(\hat{s}(t), K_E, \hat{F}(t))$ . Thus, by Lemma 3, temporal synchronization will succeed for frame  $\hat{Y}(t+1)$ . ■

**Theorem 3.4.2 (Ineffectiveness of frame insertion attack)** *Suppose  $Y(u)$  and  $Y(u+1)$  are consecutive frames of the watermarked video, and  $\hat{Y}(t) = Y(u)$ ,  $\hat{Y}(t+$*

$N + 1) = Y(u + 1)$ ,  $N > 0$ , and all frames  $\hat{Y}(t + i)$ ,  $i \in \{1, 2, \dots, N\}$  are arbitrary, unwatermarked frames inserted as an attack, and temporal synchronization is achieved for frame  $\hat{Y}(t)$ . Then, temporal synchronization will be achieved for frame  $\hat{Y}(t + N + 1)$ .

**Proof**  $\hat{Y}(t) = Y(u)$  implies  $\hat{K}(t) = K(u)$ ,  $\hat{s}(t) = s(u)$ , and  $\hat{F}(t) = F(u)$ .  $\hat{Y}(t + N + 1) = Y(u + 1)$  implies  $\hat{K}(t + N + 1) = K(u + 1)$ ,  $\hat{s}(t + N + 1) = s(u + 1)$ . Lemma 1 applies to  $s(u)$  and  $s(u + 1)$ , thus  $s(u + 1) = \phi(s(u), K_E, F(u))$ . And thus,  $\hat{s}(t + N + 1) = \phi(\hat{s}(t), K_E, \hat{F}(t))$ .

Temporal synchronization on frame  $\hat{Y}(t)$  is achieved, so by Lemma 5, the state  $\phi(\hat{s}(t), K_E, \hat{F}(t))$  will be at the head of the queue. The next  $N$  (inserted) frames of  $\hat{Y}$  after  $\hat{Y}(t)$  are all unwatermarked, which implies that temporal synchronization will not succeed for those frames (Lemma 3). By Lemma 6, however, the queue is not changed during the processing of any of the  $N$  frames and  $\phi(\hat{s}(t), K_E, \hat{F}(t))$  will remain at the head of the queue. Thus, by Lemma 3, temporal synchronization will be achieved for frame  $\hat{Y}(t + N + 1)$ . ■

**Theorem 3.4.3 (Vulnerability to frame dropping)** *Suppose  $Y(u)$ ,  $Y(u + 1)$ , and  $Y(u + 2)$  are consecutive frames of the watermarked video, watermarked using keys  $K(u) = \lambda(s(u))$ ,  $K(u + 1) = \lambda(s(u + 1))$ , and  $K(u + 2) = \lambda(s(u + 2))$ , respectively. Also assume  $s(u) \neq s(u + 1) \neq s(u + 2)$ , and  $s(u + 2) \notin \mathcal{S}_0$ . Suppose  $\hat{Y}(t) = Y(u)$ , but the next frame is dropped so that  $\hat{Y}(t + 1) = Y(u + 2)$ . Suppose  $\hat{s}(t + 1)$  is not in the detector's queue at time  $t$ . Then, temporal synchronization will not succeed for frame  $\hat{Y}(t + 1)$ , even if temporal synchronization succeeded for frame  $\hat{Y}(t)$ .*

**Proof**  $\hat{Y}(t) = Y(u)$  implies  $\hat{K}(t) = K(u)$  and  $\hat{s}(t) = s(u)$ . Likewise,  $\hat{Y}(t + 1) = Y(u + 2)$  implies  $\hat{K}(t + 1) = K(u + 2)$  and  $\hat{s}(t + 1) = s(u + 2)$ . If temporal synchronization is not achieved for frame  $\hat{Y}(t)$ , by Lemma 6, the queue will not change when the detector examines frame  $\hat{Y}(t)$ . Since  $\hat{s}(t + 1)$  was not in the queue at time

$t$ , then  $\hat{s}(t+1)$  will not be in the queue when  $\hat{Y}(t+1)$  is examined. Furthermore, since  $\hat{s}(t+1) \notin \mathcal{S}_0$ , by Lemma 3 temporal synchronization will not be achieved for frame  $\hat{Y}(t+1)$ .

Suppose temporal synchronization succeeded for frame  $\hat{Y}(t)$ . Then by Lemma 5, state  $\tilde{s} = \phi(\hat{s}(t), K_E, \hat{F}(t))$  and possibly state  $\hat{s}(t)$  will be in the queue. However neither  $\hat{s}(t) = s(u)$  nor  $\tilde{s} = s(u+1)$  is equal to  $\hat{s}(t+1) = s(u+2)$  since  $s(u) \neq s(u+1) \neq s(u+2)$ . It was assumed that state  $\hat{s}(t+1)$  was not in the queue at time  $t$ , thus when the detector examines frame  $\hat{Y}(t+1)$ , the state  $\hat{s}(t+1)$  will not be in the queue. Since  $\hat{s}(t+1) = s(u+2) \notin \mathcal{S}_0$ , by Lemma 3 temporal synchronization will not be achieved for frame  $\hat{Y}(t+1)$ . ■

From the proof of Theorem 3.4.1, it can be seen that the  $\phi(\cdot)$  function induces a “chain of states” that is created by the embedder (through the state transitions in the key generator) and traced by the detector (using the queue and state predictor). The frame dropping attack severs the chain, causing the watermark detector to be unable to discover the state sequence used by the embedder. A frame transposition attack can affect the state sequence similarly. In Section 3.5, watermark detection in the presence of frame dropping and transposition attack shall be addressed by adding temporal redundancy into the key schedule.

When the detector loses temporal synchronization, Lemma 3 shows why recovering synchronization is relatively simple for time-invariant key and time-periodic key watermarks, and difficult for time-independent key watermarks. For a time-invariant key watermark, the state of the key generator for all time is  $s(t) = \hat{s}(t) = s_{K_E} \in \mathcal{S}_0$ . Thus, synchronization will always succeed for any watermarked frame in a time-invariant key watermark, even without a queue. For a periodic key watermark, the state sequence includes  $s_{K_E} \in \mathcal{S}_0$  in each period. If the detector loses synchronization, it will be able to recover synchronization at a frame which  $\hat{s}(t) = s_{K_E}$ , or a frame which  $\hat{s}(t) \in Q$  in a future period. However, the situation is much different for a time-independent key watermark. If the detector loses synchronization, a frame that

is watermarked by the state  $s_{K_E}$  or a state in the detector’s queue may not appear until nearly  $|\mathcal{K}|$  frames in the future. Synchronization for a time-independent key watermark by search alone may not be practical, and synchronization may require embedding additional information into the watermarked signal (such as an explicit synchronization signal) to describe  $\hat{K}(t)$ .

### 3.5 Temporal Redundancy and Synchronization

Section 3.4 showed that the watermark detector may be vulnerable to frame dropping and transposition attacks. However, the resilience of the watermark detector against these attacks may be increased by adding temporal redundancy into the key schedule. Temporal redundancy adds robustness into the key schedule, permitting some frames to be dropped or re-ordered without adversely affecting temporal synchronization. A modified embedder is described which adds temporal redundancy by watermarking multiple frames of the video with an identical key.

One strategy for increasing the robustness of the watermark is to limit the effect of synchronization loss to as few frames as possible. If the detector can recover synchronization quickly, then the effect of losing synchronization is not severe. An example of this strategy is the time-periodic key watermark. If the detector loses synchronization, then a frame that allows the detector to recover synchronization should be examined by the detector in the near future.

The embedder’s key generator can be modified to produce a key schedule that is similar to, but not necessarily identical to a time-periodic key watermark. The modification entails “resetting” the key generator after  $\alpha$  consecutive frames of the video have been watermarked. When the key generator is reset, its current state is set to a member of  $\mathcal{S}_0$ , similar to the initialization which occurs before  $t = 0$ . The parameter  $\alpha$  is the *period*, although the name is somewhat a misnomer because the watermark state sequence is not necessarily strictly periodic. The larger  $\alpha$ , the less often the key generator is reset and the key schedule produced by the key generator

has less temporal redundancy. The key generator may also be reset at random intervals, where  $\alpha$  is the expected number of frames between resets. This change does not require any modification to the watermark detector, nor is it necessary to provide  $\alpha$  to the detector as side-information because temporal synchronization will succeed when a frame is watermarked with a state  $\hat{s}(t) \in \mathcal{S}_0$ . A frame watermarked when the key generator state is a member of  $\mathcal{S}_0$  is known as a *resynchronization frame*, because these frames allow the detector to recover synchronization if synchronization has been lost, as well as initial synchronization.

Theorem 3.4.3 shows that the loss of a single state (frame) can cause the detector to lose synchronization, indicating the fragileness of the state sequence. Another strategy for increasing the robustness of the watermark is to find a means for protecting the state sequence so that the loss of individual states can be tolerated without synchronization loss. Consider a modified embedder in which the key generator does not change its state after every frame of the video is watermarked. In the modified embedder, the current state of the key generator changes to  $\phi(s(t), K_E, F(t))$  only after  $\beta$  consecutive frames are watermarked, using the feature vector of the last frame to determine the next state. The current state of the state machine remains unchanged and the feature vector is ignored for all other frames during step 5 of the WEP. The parameter  $\beta$  is known as the *repeat* parameter. The resulting state sequence resembles that of Figure 3.4.

Now, it will be shown that having consecutive frames watermarked using the same key generator state will not destroy temporal synchronization at the detector. Because of this property, it is not necessary to modify the watermark detector to accommodate the change made to the embedder, or to provide  $\beta$  to the detector as side-information. Also, if the feature vectors are identical over those frames, the detector's queue will not change over those frames.

**Lemma 7 (Queue Lemma III)** *Suppose  $\hat{Y}(t)$  and  $\hat{Y}(t+1)$  are two consecutive frames watermarked using the same key  $\lambda(\hat{s}(t)) = \lambda(\hat{s}(t+1))$  and thus, same state  $\hat{s}(t) = \hat{s}(t+1)$ . Suppose temporal synchronization is achieved for frame  $\hat{Y}(t)$  and the*



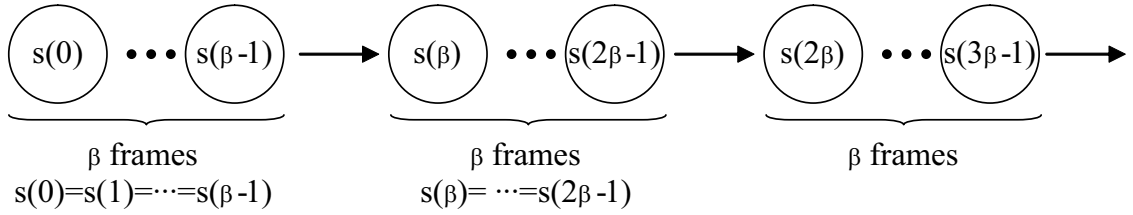


Fig. 3.4. State sequence of modified embedder. Each circle represents the state used to watermark a single video frame. Arrows indicate frames where state transitions occur in the key generator.

*queue capacity is sufficiently large. Then, temporal synchronization will be achieved for frame  $\hat{Y}(t+1)$ . Furthermore, if  $\hat{F}(t+1) = \hat{F}(t)$ , then the detector queue after examining frame  $\hat{Y}(t+1)$  is identical to that after examining  $\hat{Y}(t)$ .*

**Proof** Suppose  $\hat{s}(t) \in \mathcal{S}_0$ . Then  $\hat{s}(t+1) = \hat{s}(t) \in \mathcal{S}_0$ , so by Lemma 4, temporal synchronization will succeed for frame  $\hat{Y}(t+1)$ . By Lemma 5, the head of the queue will be  $q(0) = \phi(\hat{s}(t), K_E, \hat{F}(t))$  after examining frame  $\hat{Y}(t)$ . If  $\hat{F}(t+1) = \hat{F}(t)$ , then  $\phi(\hat{s}(t+1), K_E, \hat{F}(t+1)) = \phi(\hat{s}(t), K_E, \hat{F}(t))$ , which is already at the head of the queue and no new state will be added to the queue during step 5 of the WDP for frame  $\hat{Y}(t+1)$ . Thus, the queue after examining frame  $\hat{Y}(t+1)$  is identical to that after examining  $\hat{Y}(t)$ .

Now suppose  $\hat{s}(t) \notin \mathcal{S}_0$ . By Lemma 5, successful temporal synchronization for frame  $\hat{Y}(t)$  implies that after examining  $\hat{Y}(t)$ , the queue has  $q(0) = \phi(\hat{s}(t), K_E, \hat{F}(t))$  and  $q(1) = \hat{s}(t)$ . Consider the WDP when frame  $\hat{Y}(t+1)$  is examined by the detector. State  $\hat{s}(t+1) = \hat{s}(t)$  is in the queue, so temporal synchronization will succeed in step 3 of the WDP. During this step, the state  $\hat{s}(t)$  will be promoted and thus  $q(0) = \hat{s}(t)$  and  $q(1) = \phi(\hat{s}(t), K_E, \hat{F}(t))$ . But if  $\hat{F}(t+1) = \hat{F}(t)$  then  $\phi(\hat{s}(t+1), K_E, \hat{F}(t+1)) = \phi(\hat{s}(t), K_E, \hat{F}(t))$ , which is already in the queue, so during step 5 of the WDP, that state shall be promoted to the head and  $q(0) = \phi(\hat{s}(t), K_E, \hat{F}(t))$  and  $q(1) = \hat{s}(t)$ .

No other states in the queue are affected by the WDP. Thus, if  $\hat{F}(t+1) = \hat{F}(t)$ , then the queue is not changed when  $\hat{Y}(t+1)$  is examined. ■

**Lemma 8 (Corollary)** *Suppose  $\hat{Y}(t), \hat{Y}(t+1), \dots, \hat{Y}(t+\beta-1)$  are consecutive frames watermarked using the same state  $\hat{s}(t) = \hat{s}(t+1) = \dots = \hat{s}(t+\beta-1)$  for  $\beta > 1$ , and that temporal synchronization is achieved for frame  $\hat{Y}(t)$ . Then temporal synchronization will be achieved for frames  $\hat{Y}(t+1), \dots, \hat{Y}(t+\beta-1)$ . Furthermore, if  $\hat{F}(t) = \hat{F}(t+1) = \dots = \hat{F}(t+\beta-1)$ , then the queue after the detector examines frame  $\hat{Y}(t+\beta-1)$  is identical to the queue after the detector examines frame  $\hat{Y}(t)$ .*

**Proof** Apply Lemma 7 to every pair of frames  $\hat{Y}(t)$  and  $\hat{Y}(t+1)$ ,  $\hat{Y}(t+1)$  and  $\hat{Y}(t+2)$ ,  $\dots$ , and  $\hat{Y}(t+\beta-2)$  and  $\hat{Y}(t+\beta-1)$ . ■

**Theorem 3.5.1 (Temporal redundancy and frame dropping.)** *(See Figure 3.5 for a diagram showing the assumptions of this theorem, as the hypothesis is lengthy.) Suppose  $Y(u-1), Y(u), \dots, Y(u+\beta)$  are consecutive frames of the watermarked video produced by the modified embedder such that:*

- *Frame  $Y(u-1)$  is watermarked by key  $K(u-1) = \lambda(s(u-1))$*
- *Frame  $Y(u+\beta)$  is watermarked by key  $K(u+\beta) = \lambda(s(u+\beta))$*
- *All frames  $Y(u), \dots, Y(u+\beta-1)$ , denoted set  $\mathcal{Z}$ , are watermarked by the key  $K^* = K(u) = \dots = K(u+\beta-1) = \lambda(s^*) = \lambda(s(u)) = \dots = \lambda(s(u+\beta-1))$*
- *Feature value does not change for the frames in set  $\mathcal{Z}$ , or  $F^* = F(u) = \dots = F(u+\beta-1)$ .*
- *The state transitions of the key generator occur after watermarking frames  $\hat{Y}(u-1)$  and  $\hat{Y}(u+\beta-1)$ . Thus,  $\phi(s(u-1), K_E, F(u-1)) = s^*$  and  $\phi(s^*, K_E, F^*) = s(u+\beta)$ .*

*The watermarked video is provided to the detector, where frames  $\hat{Y}(t-1) = Y(u-1)$  and  $\hat{Y}(t+\tau) = Y(u+\beta)$ , fixed  $\tau \leq \beta$ , are not attacked. However, one or more*

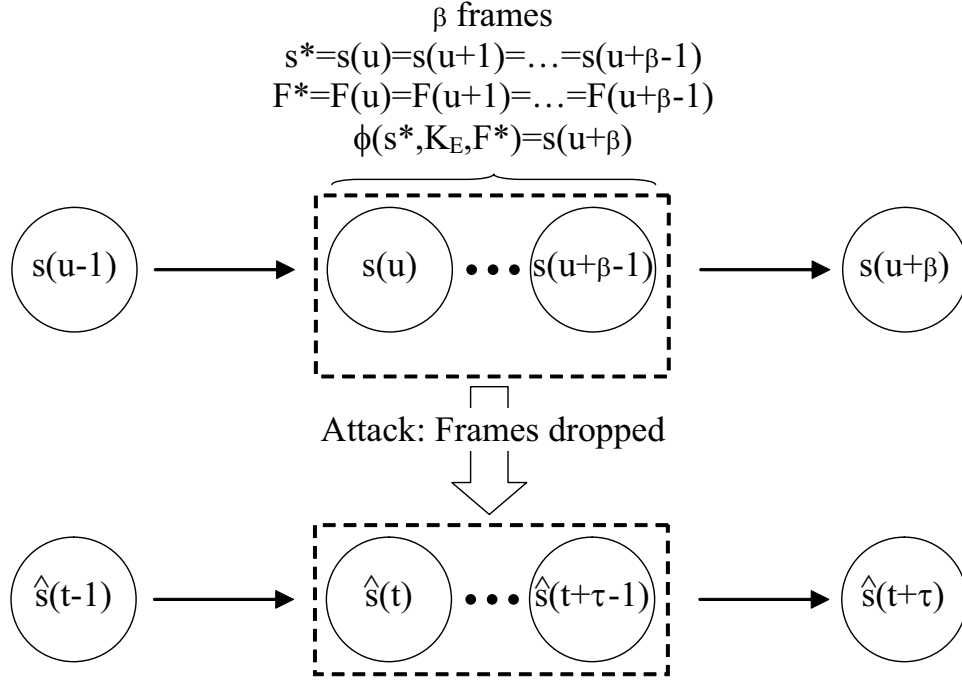


Fig. 3.5. State sequence in a frame dropping attack

frames in set  $\mathcal{Z}$  may be dropped. Let the set  $\hat{\mathcal{Z}} = \{\hat{Y}(t), \dots, \hat{Y}(t+\tau-1)\}$  be the set of frames from  $\mathcal{Z}$  that remain after the frame drop attack. Assume there are no other attacks on  $\hat{\mathcal{Z}}$ , and temporal synchronization is achieved for frame  $\hat{Y}(t-1)$ . Then, if there is at least one frame in  $\hat{\mathcal{Z}}$ , then temporal synchronization will be achieved for all the frames in  $\hat{\mathcal{Z}}$  as well as frame  $\hat{Y}(t+\tau)$ .

**Proof** Temporal synchronization is successful for frame  $\hat{Y}(t-1)$ . Thus, the queue shall contain the state  $\phi(\hat{s}(t-1), K_E, \hat{F}(t-1)) = \phi(s(u-1), K_E, F(u-1)) = s^*$  (Lemma 5). Since  $\hat{\mathcal{Z}}$  is non-empty and no other attacks have been performed in  $\hat{\mathcal{Z}}$ , there exists a frame,  $\hat{Y}(t) \in \hat{\mathcal{Z}}$ . For any frame in  $\hat{\mathcal{Z}}$ , the state of the key generator used to watermark the frame is  $s^*$  and the feature vector extracted from the frame is  $F^*$ . Since  $s^*$  is in the queue, temporal synchronization will be achieved for frame  $\hat{Y}(t)$  (Lemma 3). After examining  $\hat{Y}(t)$ , the head of the queue will be the state  $\phi(\hat{s}(t), K_E, \hat{F}(t)) = \phi(s^*, K_E, F^*) = s(u+\beta) = \hat{s}(t+\tau)$  (Lemma 5). If there is more

than one frame in  $\hat{\mathcal{Z}}$ , temporal synchronization will be achieved for every frame in  $\hat{\mathcal{Z}}$  and the queue will remain unchanged (Lemma 8). Thus, when the detector examines frame  $\hat{Y}(t + \tau)$ , the state  $\hat{s}(t + \tau)$  will be in the queue and by Lemma 5, temporal synchronization will be achieved for  $\hat{Y}(t + \tau)$ . ■

Theorem 3.5.1 shows that the watermark produced by modified embedder, in which sets of  $\beta$  consecutive frames are watermarked by the same state of the key generator, will be resilient against frame dropping attack unless all  $\beta$  frames in a set are dropped (assuming feature vectors are constant over each set, or if feature extraction is not used). By performing state transitions every  $\beta$  frames instead of at every frame as described in Section 3.2, redundant copies of each state are created prior to a state transition. These redundant copies will not confuse the detector (Lemma 8), and the detector only needs to observe each state once to maintain temporal synchronization (Theorem 3.5.1). Any frame dropping attack which drops less than  $\beta$  consecutive frames will fail to desynchronize the detector.

Lemma 8 also demonstrates that temporal upsampling, where the attacked video is obtained by repeating each frame of the watermarked video multiple times, will not desynchronize or otherwise affect the watermark detector. In fact, temporal upsampling increases the temporal redundancy of the watermark, making the watermark more robust against frame dropping.

Both the strategies described above require the watermark embedder to be modified from Section 3.2, by the insertion of a *Temporal Redundancy Control*. The new embedder is shown in Figure 3.6. The temporal redundancy control interfaces with the WEP, as follows:

1. *Initialization*: Set counters  $a = 0$  and  $b = 0$ . The key generator is reset, by setting the current state to a member of  $\mathcal{S}_0$ .
2. During step 1 of the WEP, the temporal redundancy control provides  $\lambda(s(t))$  to the watermark signal generator, where  $s(t)$  is the current state of the key generator.

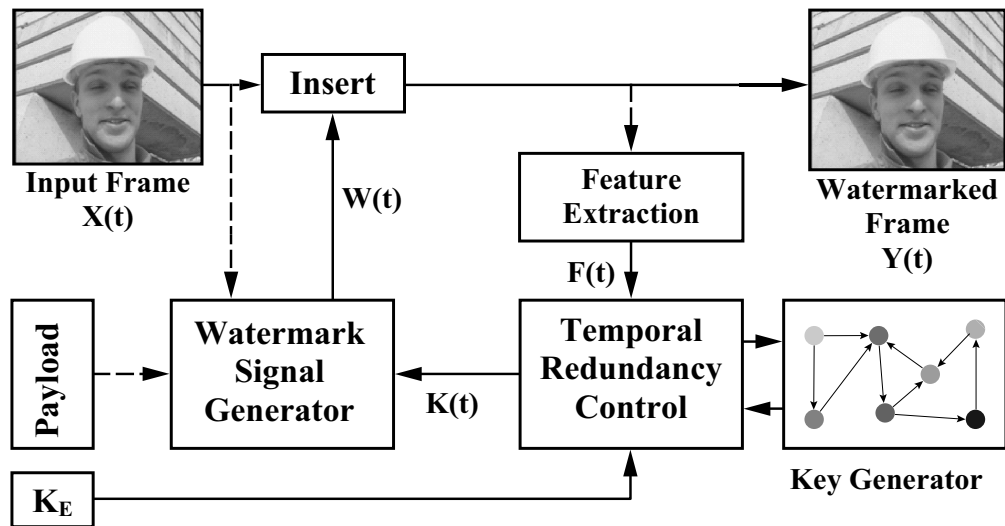


Fig. 3.6. Modified embedder with temporal redundancy control

3. During step 5 of the WEP, the temporal redundancy control increments  $a$  and  $b$ . Then:
  - (a) If  $a = \alpha$ , then the key generator and the temporal redundancy control are reset for the next frame of the video. Return to step 1 for the next frame of the video.
  - (b) Otherwise, if  $b = \beta$ , then the key generator performs a state transition. Set  $b = 0$ , and then return to step 2 for the next frame of the video.
  - (c) Otherwise, return to step 2 for the next frame of the video, without changing the current state of the key generator.

Selecting  $\alpha = \infty$  and  $\beta = 1$  produces a watermark key schedule identical to that of Section 3.2.

### 3.5.1 Adaptive state transitions

The temporal redundancy control described above uses two parameters to produce key sequences of varying degrees of temporal redundancy. The period ( $\alpha$ ) is the number of frames watermarked before the key generator is reset, which corresponds to the interval between resynchronization frames. The repeat ( $\beta$ ) is the number of consecutive frames watermarked with the same key before the key generator is used to produce a new key. Generally, decreasing  $\alpha$  and increasing  $\beta$  increases the amount of temporal redundancy in the key sequence and provides increased robustness against attack. However, using a fixed  $\beta$  for the temporal redundancy control can sometimes cause a dramatic loss in temporal redundancy when feature vectors change.

A key assumption in Theorem 3.5.1 is that feature vectors remain constant over each set of  $\beta$  frames watermarked with the same state. However, the actual state transition of the embedder's key generator uses only the feature vector of the last frame in a set. If feature vectors change within a set of  $\beta$  frames, then temporal re-

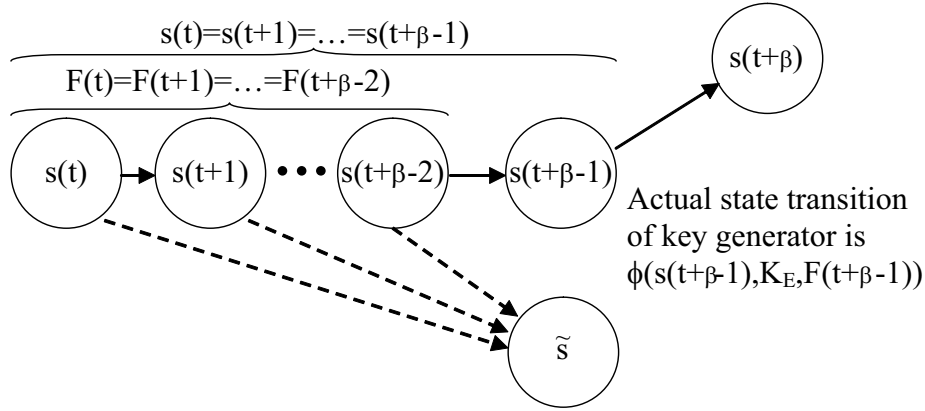


Fig. 3.7. Example of changing feature values reducing temporal redundancy

dundancy is reduced. An example is illustrated in Figure 3.7. Frames  $Y(t), \dots, Y(t + \beta - 1)$  are watermarked with the same state  $s^*$ , and  $Y(t), \dots, Y(t + \beta - 2)$  have the same feature vector  $F^*$ . Assume that  $\phi(s^*, K_E, F^*) = \tilde{s}$ , which is some state in the state machine. However, suppose that the feature vector changes value for frame  $F(t + \beta - 1)$ , and that  $\phi(s^*, K_E, F(t + \beta - 1)) = s(t + \beta) \neq \tilde{s}$ . Then, the frames  $Y(t), \dots, Y(t + \beta - 2)$  will not provide the detector queue with the state  $s(t + \beta)$ , even if temporal synchronization is successful for those frames. Because of the change in the feature value, the loss of frame  $Y(t + \beta - 1)$  would be catastrophic for temporal synchronization at the detector.

An alternate strategy is to perform state transitions in the key generator based on when the feature values change, instead of a fixed interval of  $\beta$  frames. That is, the key generator changes its current state only after (1) feature values have been constant for  $\beta$  consecutive frames, and (2) at least  $\beta$  frames have been watermarked with the same state. This strategy adapts the key generator state transitions to the characteristics of the video to increase the temporal redundancy of the key sequence and improve the robustness of the watermark.

### 3.5.2 Security

Temporal redundancy is an advantage for temporal synchronization of the watermark detector, however redundancy is a disadvantage for security. A watermark whose key sequence has more redundancy is less random and is more vulnerable to estimation attacks. Thus, the state transitions should appear as random as possible, which increases the difficulty for an attacker to predict the key schedule (or equivalently, the state sequence).

One method for increasing the degree of randomness in the key schedule is to use a non-deterministic state machine for the key generator. The key schedule produced by a deterministic key generator is dependent only on the embedding key and the feature vectors obtained from the video. The state transitions and initial states in a non-deterministic state machine introduce randomness into the key sequence. Even a small degree of randomness for each state transition can have a large effect on the state sequence because each state transition affects not just the next state of the state machine, but potentially all future states of the state machine.

Another method for increasing the randomness in the key schedule is to define the state transition function by using cryptographic hash functions. A cryptographic hash function, or a one-way hash function,  $H(M)$  is a function that accepts as input an arbitrary length binary string message  $M$  (in this context,  $M$  is not related to the watermark payload) and produces a fixed length bit string output of size  $L$  bits, known as a digest or hash value [16, 292]. The notation  $H(x_1, x_2, \dots)$  is used to indicate the  $L$ -bit digest produced by  $H$  when the concatenation of the binary representation of the values  $x_1, x_2, \dots$  is provided as the message input. Given a message  $M$ , it is very easy to obtain digest  $H(M)$ . However, given the digest, one cannot easily construct a message that hashes to that digest. Cryptographic hash functions are typically used for constructing message authentication codes [16].

An example state transition function using hash functions is:

$$s(t + 1) = \phi(s(t), K_E, F(t)) = s_{H(s(t), K_E, F(t))} \quad (3.6)$$



where the set of states is  $\mathcal{S} = \{s_0, s_1, s_2, \dots, s_{2^L-1}\}$  with cardinality  $|\mathcal{S}| = 2^L$ . An example state transition function for a non-deterministic state machine is:

$$s(t+1) = \phi(s(t), K_E, F(t)) = s_{H(s(t), K_E, F(t), \Gamma(t))} \quad (3.7)$$

where  $\Gamma(t)$  is a randomly selected member of the set  $\{0, 1, 2, \dots, 2^R - 1\}$ ,  $R$  a fixed positive integer. Because the detector must insert into the queue and search all  $2^R$  possible values of  $\phi(\cdot)$ ,  $R$  is usually chosen to be a small number.

The cryptographic hash function increases the difficulty in predicting or identifying the state sequence. First, the hash function allows  $K_E$  to be involved in each state transition, which is unlike a PRNG in which there is only a single state transition sequence and  $K_E$  merely chooses the seed (the starting state). The hash is similar to having a different PRNG for each  $K_E$ . The properties of the hash function make it difficult to predict the state sequence without knowledge of  $K_E$ , and obtaining  $K_E$  from observation of the state sequence is difficult. A good hash function also allows the next state to be determined while avoiding preferential treatment to the binary values of particular parameters, such as the current state, feature values, and random bits.

The major security concern lies in the resynchronization frames. These frames may occur frequently over the video, which presents an opportunity for estimation attacks similar in spirit to those attacking a time-periodic key watermark. The frequency of resynchronization frames (parameter  $\alpha$ ) is a trade-off between the ease of synchronization and security. To improve security, resynchronization frames must be more difficult to identify and estimate. One method to do this is to use image-dependent watermarking [89, 293]. Another method is to use the feature vector of the previous frame to determine the key for the resynchronization frame. That is, if the key generator is reset for watermarking frame  $X(t)$ ,  $s(t)$  is not set to a member of  $\mathcal{S}_0$ , but to a state which is dependent on  $F(t-1)$ ,  $K_E$ , and possibly  $\Gamma(t)$ . The WDP (step 2) would be modified accordingly. These methods generate different watermark patterns for the resynchronization frames and make them more difficult to identify and estimate. Neither of these methods are implemented here.

### 3.6 Experimental Results

To evaluate the effectiveness of adding temporal redundancy on temporal synchronization, the watermark embedding and detection models have been implemented for uncompressed video sequences. The implementations of the watermark signal generation, insertion, and signal detection techniques are simple, which maintains the focus on the temporal structure of the watermark and its effects on synchronization (and not on the intricacies of signal embedding and detection.)

The watermark signal generator is a PRNG that is seeded by  $K(t)$  for each video frame  $X(t)$ , producing a zero-mean unit variance i.i.d. Gaussian distributed pseudo-random sequence  $W(t)$ . The watermark embedder inserts  $W(t)$  additively in the spatial domain (pixel values) of the video frame, producing watermarked  $Y(t)$ . Perceptual shaping is not used. Only the luminance of the video was watermarked; the embedder ignores the chrominance.

The watermark signal detector applies a spatial de-correlating filter to reduce host-signal interference (see [38] and [3,109]), followed by correlation and comparison with a threshold value. The decorrelating filter is applied by convolving the input image with the point-spread function

$$h = \frac{1}{4} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}. \quad (3.8)$$

Correlation is similar to (2.3), except that the dot product is with the filtered input:

$$\rho(t) = \frac{1}{N} [Z(t) * h] \cdot W_K \quad (3.9)$$

where  $Z(t)$  is the input frame,  $h$  is the filter of (3.8),  $N$  is the length of the watermark signal (which is identical to the number of pixels in a video frame), and  $W_K$  is the watermark signal when key  $K$  is provided to the signal detector. The watermark embedding strength is such that the nominal correlation value is  $\rho(t) = 2.0$  when the watermark is present. The detection threshold was such that the watermark

detector reports the watermark is present in the frame when  $\rho(t) \geq T = 1.0$ . The detector queue capacity for all experiments is fixed to 10 elements.

Watermark embedders using three different key generators are examined. All three key generators include the temporal redundancy control shown in Figure 3.6 and the state transition functions are cryptographic hash functions as described in Section 3.5.2. SHA-1 [294] is used as the hash function, which produces a message digest of  $L = 160$  bits in size. The key generators also share in common  $\mathcal{K} = \{0, 1, 2, \dots, 2^{160} - 1\}$ ,  $\mathcal{S} = \{s_0, s_1, \dots, s_{2^{160}-1}\}$ ,  $\mathcal{S}_0 = s_{K_E}$ , and  $\lambda(s_i) = i$ .

The first key generator (“Features Only”) uses feature extraction and a deterministic state machine (3.6). The feature extraction produces a feature vector by partitioning each frame into non-overlapping blocks of  $32 \times 32$  pixels in size. Each block is pseudo-randomly assigned to one of three groups,  $\Omega_1, \Omega_2, \Omega_3$ , using  $K_E$  as the seed to determine the assignment. This assignment is determined once, and the same block assignments is used for each frame of the video. The feature vector is determined by

$$F(t) = \left\langle \left\lfloor \frac{\text{mean}(\Omega_1)}{\Delta} \right\rfloor, \left\lfloor \frac{\text{mean}(\Omega_2)}{\Delta} \right\rfloor, \left\lfloor \frac{\text{mean}(\Omega_3)}{\Delta} \right\rfloor \right\rangle \quad (3.10)$$

where  $\text{mean}(\Omega_i)$  is the mean of all pixels values in  $Z(t)$  assigned to group  $\Omega_i$  and  $\lfloor \cdot \rfloor$  indicates the floor function (or rounding towards zero.) The division by  $\Delta$  and floor effectively implement a uniform scalar quantizer with step size  $\Delta$ . In the experiments,  $\Delta = 4$  was used. With each pixel value in the range  $0 \dots 255$ ,  $\Delta = 4$  implies that each feature in  $F(t)$  is in the range  $0 \dots 64$ , or a 6-bit integer value. These features are selected because they are straightforward to implement and computationally efficient to obtain, and not because they are optimal in any sense (including robustness and security).

The second key generator (“Random Only”) uses the non-deterministic key generator with the state transition function of (3.7), with  $R = 2$ . The “Random Only” embedder does not perform feature extraction. The third key generator (“Adaptive”) uses a non-deterministic key generator with the state transition function of

(3.7), with  $R = 2$ , the same features as the “Features Only” key generator, and the adaptive state transitions described in Section 3.5.1.

The results show the mean performance of the detector after 10 trials of each of the *Akiyo*, *Foreman*, and *Bus* sequences ( $352 \times 288$  CIF, 30 frames/sec). The *Akiyo* and *Foreman* sequences are 300 frames in length while the *Bus* sequence is 150 frames in length. A random  $K_E$  was used for each trial. Because the watermark detector performs an independent watermark detection attempt for each video frame, performance can be measured by the percentage of the watermarked frames that are detected in the attacked video as the temporal redundancy is varied. A detection rate of 100% indicates that the watermark detector detects the watermarks embedded in every frame of the attacked video and the attack is ineffective. A detection rate of 0% indicates that the detector is unable to detect the watermark in the attacked video. Temporal redundancy is expressed in terms of  $\alpha$  (period) and  $\beta$  (repeat) parameters.

Figure 3.8 shows the watermark detection performance after frame dropping attack, where each frame of the watermarked video is dropped with probability  $P = 0.25$  and  $0.5$ . For fixed  $\alpha$ , the detections improve for all the embedders as  $\beta$  is increased, showing that robustness improves with increasing temporal redundancy (Theorem 3.5.1). When there is little temporal redundancy ( $\beta = 1$ ), the performance of all the embedders is poor, in agreement with Theorem 3.4.3. However, the performance of the embedders improve significantly with added temporal redundancy ( $\beta = 5, 10$ ), particularly for the “Random Only” and “Adaptive” embedders. For fixed  $\beta$ , the performance improves with decreasing  $\alpha$ . When the detector loses temporal synchronization, it will not recover synchronization until it discovers a frame watermarked with a state  $\hat{s}(t) \in \mathcal{S}_0$  or in the queue. When  $\alpha$  is decreased, frames are watermarked from a state in  $\mathcal{S}_0$  more frequently, allowing the detector to recover synchronization.

To show that the loss of temporal synchronization occurs when  $\beta$  consecutive frames are dropped, the detection rate was examined after temporal decimation

attack. The results are shown in Figures 3.9 and 3.10. Decimation by a factor of  $\lambda$  retains the first frame out of every  $\lambda$  consecutive frames. Choosing to retain the first frame of every  $\lambda$  frames provides the most advantageous situation with respect to the watermark detector (and not the attacker). This shows Theorem 3.5.1 most vividly, as the intent is to show that the detection rates drop significantly when the decimation factor exceeds  $\beta$ , even in the best case. If the decimation attack retained the last frame of every  $\lambda$  frames, initial synchronization will fail when  $\lambda$  exceeds  $\beta$ .

The effect of changing feature values on temporal redundancy (described in Sec. 3.5.1) is dramatic, resulting in the poor performance of the “Features Only” embedder. For the “Random Only” and “Adaptive” embedders, the watermark detection rate is 100% when the decimation factor is equal to or below  $\beta$ , demonstrating that the loss of  $\beta - 1$  frames will not destroy temporal synchronization. The detection rate decreases significantly when the decimation factor exceeds  $\beta$ . There is one exception, in which the detection rate of these embedders when  $\alpha = 30$  and  $\beta = 5$  remains good even after the watermarked video is decimated by factor of 6. This arises because the key generator is reset so often that the watermark detector discovers a frame watermarked by a state in  $\mathcal{S}_0$  before the decimation attack can affect detection.

The insertion of unwatermarked frames into the video does not affect watermark detection, in agreement with Theorem 3.4.2. The observed detection rate is 100% for all embedders and temporal redundancy parameters. In this attack, an arbitrary number of frames from the original video are inserted between consecutive frames of the watermarked video. Failure to detect the watermark in the inserted frames does not penalize the performance of the detector because the inserted frames are not watermarked.

Frame transposition has a similar effect to frame dropping on disrupting the “chain of states” induced by  $\phi(\cdot)$ , with the difference being that the attack displaces the video frames in time instead of removing frames from the video. The implementation of this attack is as follows: The watermarked video is scanned from beginning to end, with each frame having a fixed probability ( $P = 0.25$  and  $0.5$ ) of being

interchanged with another (target) frame in the local neighborhood of the candidate frame. A temporally local frame was chosen because these frames are likely to be similar. Transposing temporally distant frames does not make much sense as an attack because these frames are generally disparate, and transposing them causes the quality of the attacked video to be severely degraded (by flickering.) The target frame is selected by generating a Gaussian random number (with variance 5.0), dropping fractions, and treating the number as a relative time index where 0 is the current frame,  $-1$  is one frame in the past,  $+1$  is one frame in the future, and so on. Because transposing a frame with itself does not accomplish anything, another random number is chosen if the relative time index is zero or out of range of the video. The performance of the watermark detection under frame transposition attack, shown in Fig. 3.11, shows similar trends to frame dropping.

Temporal frame averaging was also investigated using several moving window sizes. Figures 3.12 and 3.13 show the detection rate of the watermark after frame averaging using window sizes of  $\lambda = 2, 3,$  and  $4$ . Frame averaging does not change the sequence of frames appearing in the video, but averaging can detrimentally affect the detection rate for two reasons: First, frame averaging itself may attenuate or remove the embedded watermark, causing the detector to miss. Second, frame averaging can cause the detector to obtain different feature values than those used by the embedder to generate the key schedule. When this occurs, the detector's state prediction will fail and the detector will lose synchronization. Thus, frame averaging is interesting in that it is both a removal attack as well as a synchronization attack. The performance of the "Random Only" embedder shall be useful in explaining the watermark detection performance under the frame averaging attack because this embedder does not use feature extraction.

Attenuation of the watermark causes the watermark detector to miss for all of the embedders when  $\beta = 1$ . In particular, the poor performance of the "Random Only" embedder shows that changing feature values is not the culprit. The poor performance of  $\beta = 1$  watermarks is in agreement with other works discussing temporal

collusion attacks [82, 89]. However, watermark attenuation is also the cause of the decreased detection rates for the “Random Only” embedder for  $\beta > 1$  and  $\lambda = 2, 4$ . The reason why the “Random Only” watermark detector does not miss for window size  $\lambda = 3$  but misses for  $\lambda = 2, 4$  is not surprising once frame averaging is examined more closely. Section 3.6.1 explores frame averaging as a removal attack in detail.

Frame averaging can also cause synchronization loss by changing the values of the features. The change in feature values has a severe effect on synchronization of the “Features Only” embedder when  $\beta > 1$ . The “Adaptive” embedder often shows much better detection rate than “Features Only”, because state transitions do not occur as frequently for the “Adaptive” embedder compared with the “Features Only” embedder for fixed  $\beta$ . Because less state transitions occur for the “Adaptive” embedder, there is less opportunity for the detector to lose synchronization.

Comparing the three embedders, the “Features Only” embedder often has worse performance than the “Random Only” and the “Adaptive” embedders, whose performances are often similar. This vividly shows the effect of changing feature values on the temporal redundancy of the watermark described in Sec. 3.5.1, as the “Features Only” and “Adaptive” embedders use the same feature extractor but the performance of “Adaptive” is better than “Features Only”. Adaptive state transitions significantly improve the performance of the watermark when feature extraction is used.

The inserted watermarks are generally not noticeable in these experiments (the embedding is such that the PSNR between the watermarked and original videos is  $\geq 40$  dB), however the various attacks performed has a noticeable impact on the quality of the attacked video. The frame dropping attack is generally unnoticeable when approximately 5%–10% frames are dropped, however higher drop rates make the video appear “jerky”. Decimation reduces the frame rate, which is noticeable even when the decimation factor is 2. The frame transposition attack produces “jerky” video like the frame dropping attack, and frame averaging noticeably blurs the video.

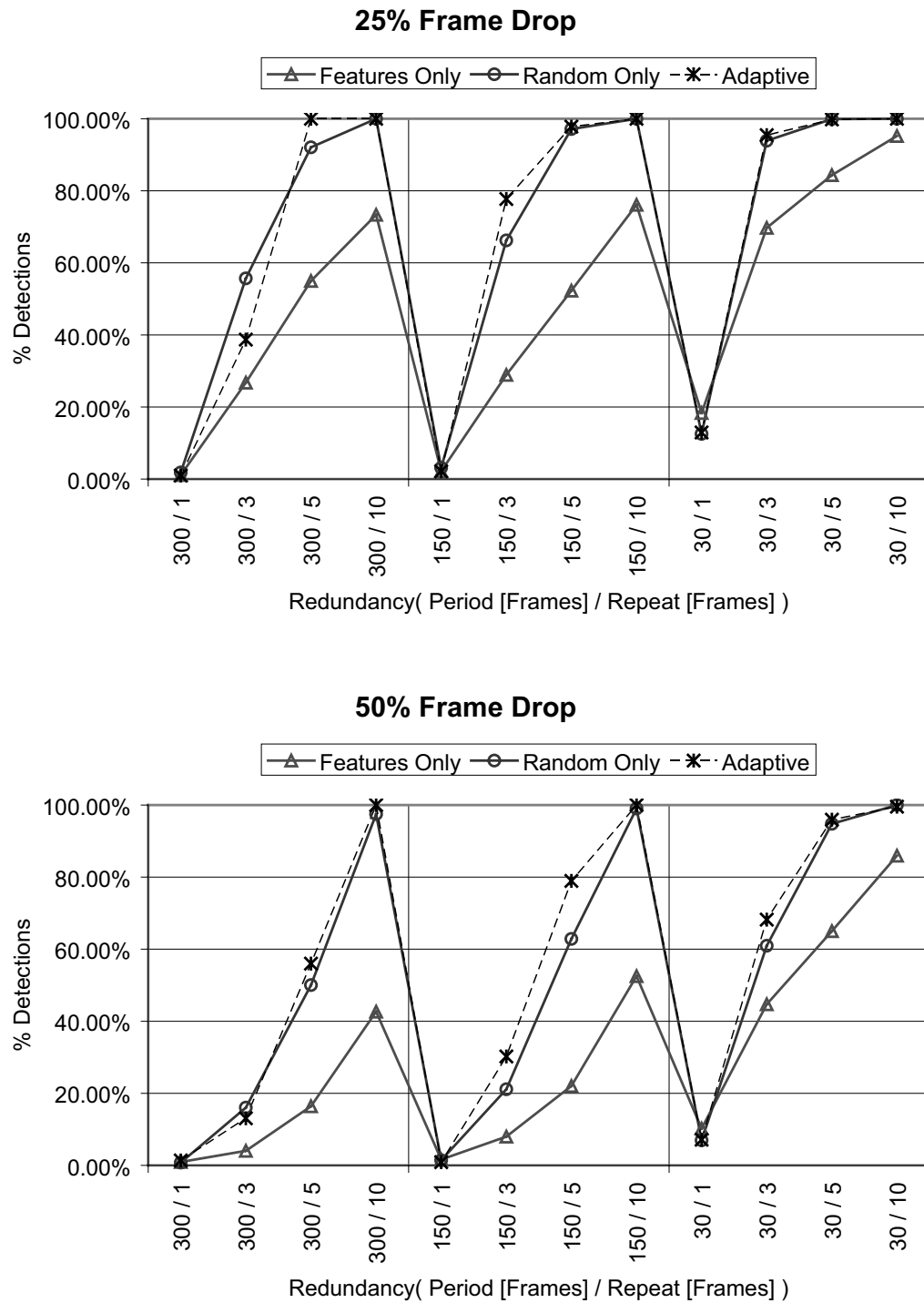


Fig. 3.8. Detection rate under frame dropping attack





Fig. 3.9. Detection rate under frame decimation attack

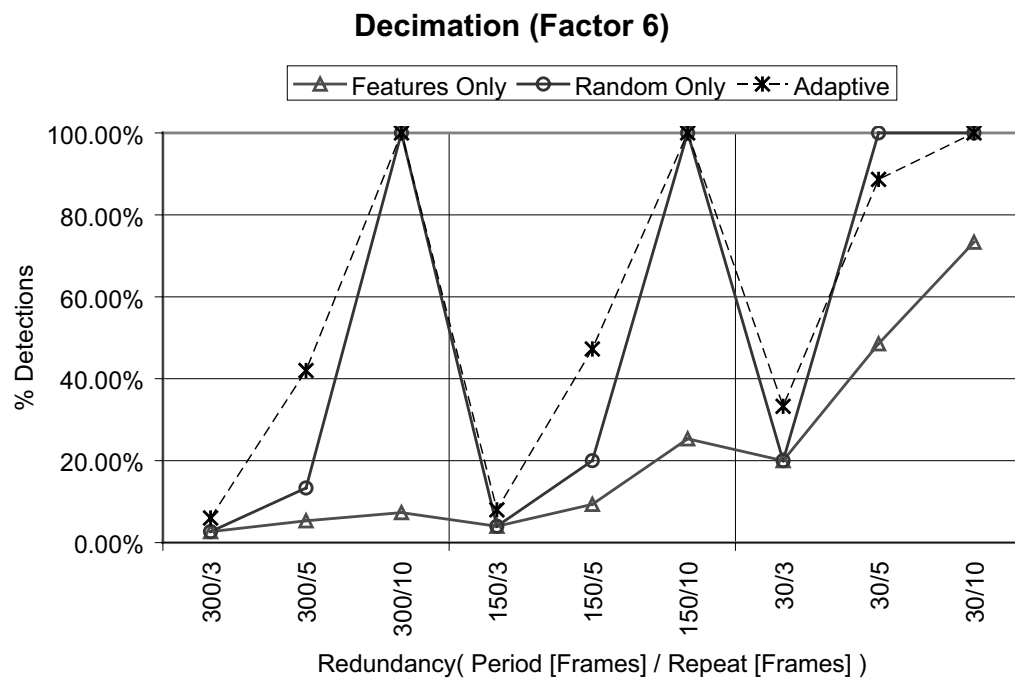
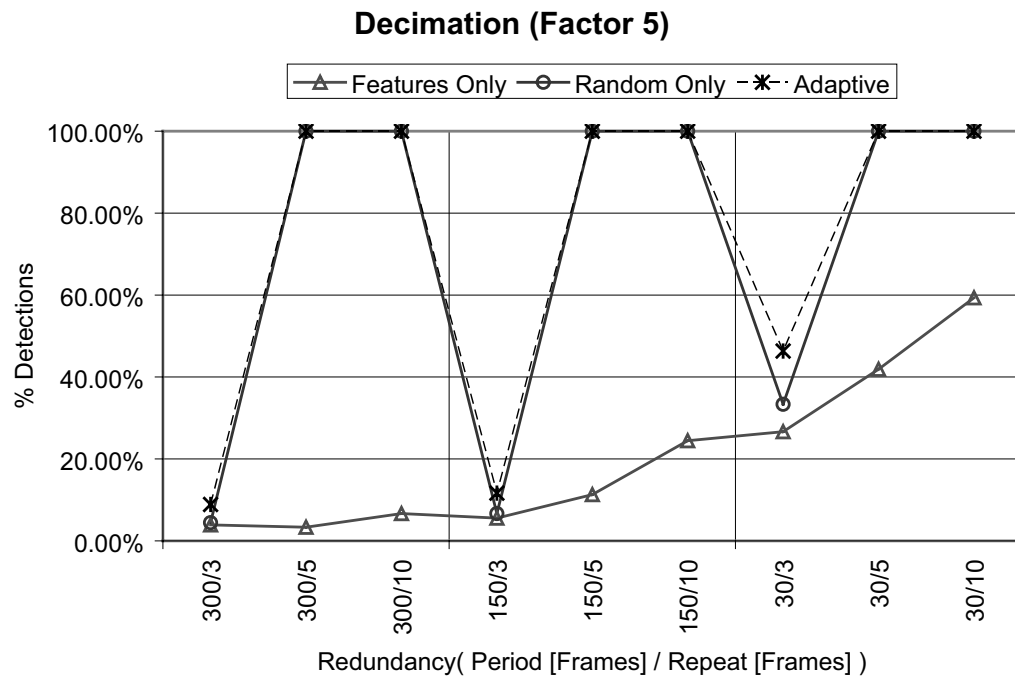


Fig. 3.10. Detection rate under frame decimation attack (continued)

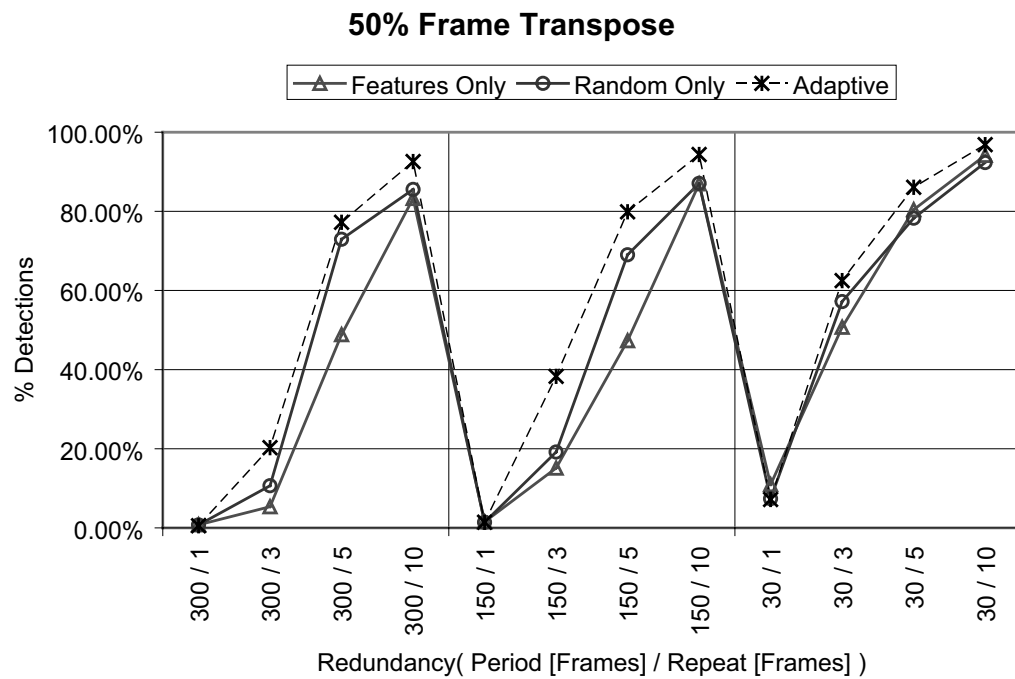
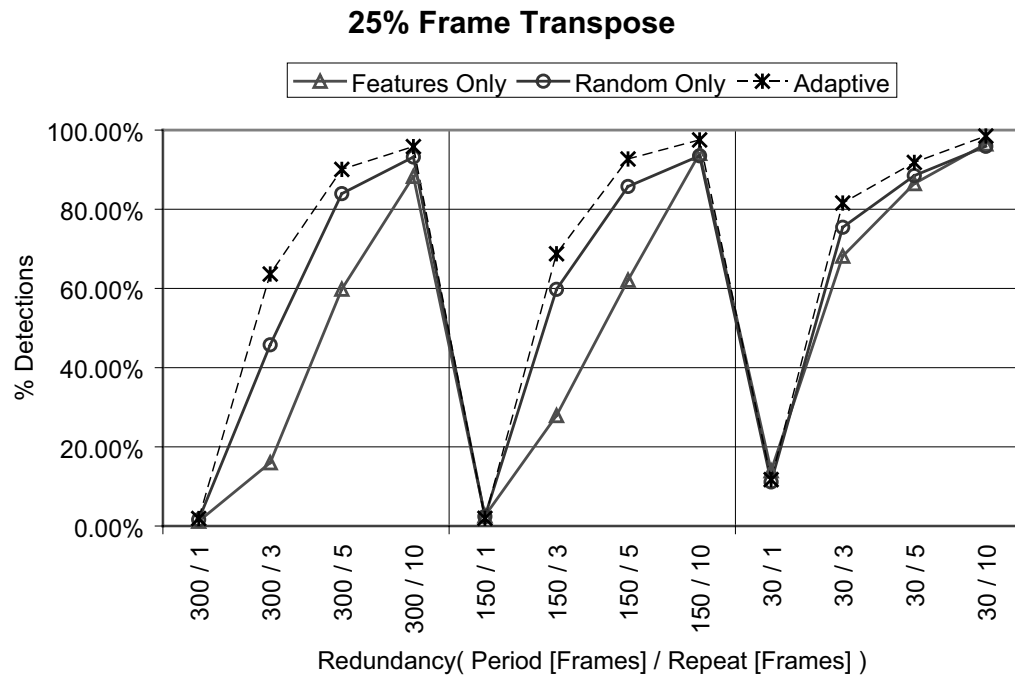


Fig. 3.11. Detection rate under frame transposition attack



Fig. 3.12. Detection rate under frame averaging attack

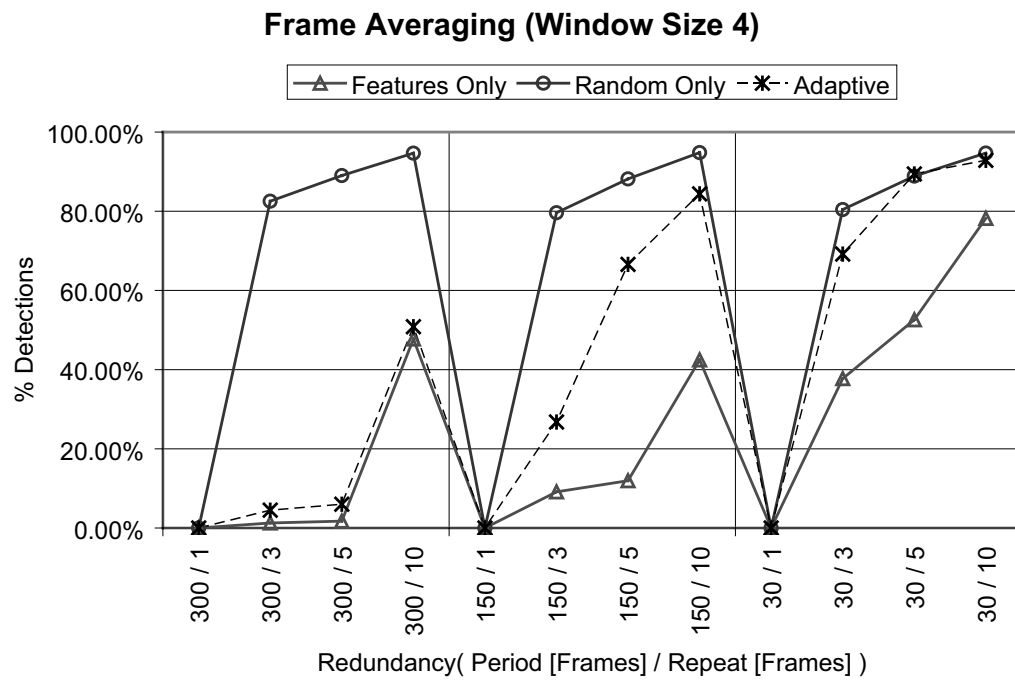


Fig. 3.13. Detection rate under frame averaging attack (continued)

### 3.6.1 Frame averaging

The earlier discussion regarding frame averaging (and Figures 3.12 and 3.13) noted that the watermark detector did not miss for the “Random Only” embedder for  $\beta > 1$  and a moving window size of  $\lambda = 3$ , but some misses were noted for  $\lambda = 2$  or  $\lambda = 4$ . This has to do with how frame averaging affects the watermark as a removal attack, as will be detailed in this section.

Before discussing frame averaging, a small digression into the correlation-based detector is worthwhile. The blind correlation-based watermark signal detector is provided with an input video frame  $Z(t)$  and key  $K$ . For simplifying this discussion, assume that the filtering of (3.8) was not performed prior to correlation. The detector generates the zero-mean candidate watermark signal  $W_K$  using  $K$  and then performs correlation as in (2.3):

$$\rho(t) = \frac{1}{N} (Z(t) \cdot W_K). \quad (3.11)$$

For a watermark that is embedded additively as

$$Y(t) = X(t) + \gamma W(t) \quad (3.12)$$

and when  $Y(t)$  is provided to the detector, or  $Z(t) = Y(t)$ , then the correlation is similar to (2.6), or

$$\rho(t) = \frac{1}{N} (X(t) \cdot W_K) + \frac{\gamma}{N} (W(t) \cdot W_K). \quad (3.13)$$

Taking expectation with respect to  $W_K$ ,

$$\mathbb{E}[\rho(t)] = \frac{1}{N} \mathbb{E}[X(t) \cdot W_K] + \frac{\gamma}{N} \mathbb{E}[W(t) \cdot W_K]. \quad (3.14)$$

The expected value of the first term (the host interference term) is zero since the original signal  $X(t)$  is expected to be uncorrelated with *any*  $W_K$ . For the second term, watermark signals corresponding to distinct keys are generally uncorrelated, thus the expected value is

$$\mathbb{E}[W_K \cdot W(t)] = \begin{cases} W(t) \cdot W(t) & W_K = W(t) \\ 0 & \text{Any other } W_K \end{cases} \quad (3.15)$$

and the expected value of the overall correlation is

$$\mathbb{E}[\rho(t)] = \begin{cases} \rho^* = \frac{\gamma}{N} [W(t) \cdot W(t)] & W_K = W(t) \\ 0 & \text{Any other } W_K \end{cases} \quad (3.16)$$

The watermark detector reports that the watermark detector is present in  $Z(t)$  if the correlation  $\rho(t)$  is larger than a threshold. Assume for this discussion that the threshold is taken to be  $T = \frac{1}{2}\rho^*$ , or that

$$\text{Detector Output} = \begin{cases} \rho(t) \geq \frac{1}{2}\rho^* & \text{Watermark Present} \\ \text{otherwise} & \text{Watermark Not Present} \end{cases} \quad (3.17)$$

Section 3.6 had mentioned that the watermark is constructed such that the nominal correlation value is  $\rho^* = 2.0$ , and the detection threshold was  $T = \frac{1}{2}\rho^* = 1.0$  for the experiments.

Temporal frame averaging is an attack which constructs an attacked frame by averaging the pixel values of the frames in the averaging window. More precisely, the frame averaging attack constructs the attacked frame  $\hat{Y}(t)$  by

$$\begin{aligned} \hat{Y}(t) &= \frac{1}{\lambda} \left[ \sum_{i=0}^{\lambda-1} Y(t+i) \right] \\ &= \frac{1}{\lambda} \left[ \sum_{i=0}^{\lambda-1} X(t+i) + \gamma W(t+i) \right] \end{aligned} \quad (3.18)$$

The frames of the original video will not be identical unless the video is static in the window. However, the video is usually highly correlated (for a sufficiently small window) and the approximation

$$X(t) \approx X(t+1) \approx \dots \approx X(t+\lambda-1) \quad (3.19)$$

is convenient. Then, (3.18) can be written as

$$\hat{Y}(t) \approx X(t) + \frac{\gamma}{\lambda} \left[ \sum_{i=0}^{\lambda-1} W(t+i) \right] \quad (3.20)$$

When the watermark detector examines  $\hat{Y}(t)$ , the correlation obtained is

$$\rho(t) = \frac{1}{N} (X(t) \cdot W_K) + \frac{\gamma}{N\lambda} \left[ \sum_{i=0}^{\lambda-1} W(t+i) \cdot W_K \right] \quad (3.21)$$

and taking expectation with respect to  $W_K$ ,

$$E[\rho(t)] = \frac{\gamma}{N\lambda} \left[ \sum_{i=0}^{\lambda-1} E[W(t+i) \cdot W_K] \right] \quad (3.22)$$

Consider averaging with a window size of  $\lambda = 2$  and a watermark constructed with any value of  $\beta$ . Suppose the averaging window is positioned such that the first frame of the window is watermarked with  $W(t) = W_A$  and the second frame is watermarked with  $W(t+1) = W_B$ , with  $W_A \neq W_B$ . (This occurs every  $\beta$  frames.) Then substituting in (3.22),

$$E[\rho(t)] = \frac{\gamma}{2N} [E[W_A \cdot W_K] + E[W_B \cdot W_K]] \quad (3.23)$$

If  $W_K$  is either  $W_A$  or  $W_B$ , then the expected correlation becomes

$$E[\rho(t)] = \frac{\gamma}{2N} [W_K \cdot W_K] = \frac{1}{2}\rho^* \quad (3.24)$$

Comparing (3.24) with (3.17), the expected correlation is identical to the threshold. This situation results in a large probability of detection miss.

For frame averaging using  $\beta = 1$  and any window size  $\lambda > 1$ , there is no temporal redundancy in the watermark so the first miss is immediately fatal. When the detector examines a watermarked frame and misses, the queue is not updated and the detector will not be able to detect the watermark in subsequent frames until a resynchronization frame is encountered. Thus, the  $\beta = 1$  detection performance is very poor for “Random Only” (as well as the other embedders) in Figures 3.12 and 3.13 regardless of the window size.

Now consider when  $\lambda = 2$  but with  $\beta \geq 2$ . State transitions occur every  $\beta$  frames, which imply that for every set of  $\beta$  frames, there are  $\beta - 1$  frames in which the averaging window contains two frames that have the same embedded watermark, and 1 frame where the averaging window contains frames with different watermarks. In the first  $\beta - 1$  frames, averaging does not affect the expected correlation value of  $E[\rho(t)] = \rho^*$ . Thus, the watermark will be detected in these frames, which also provides the next states in the detector’s queue for detecting the embedded



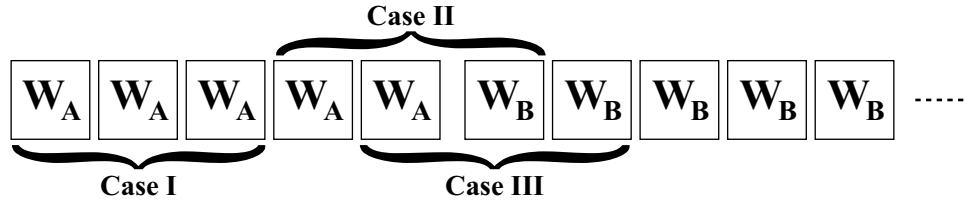


Fig. 3.14. Frame averaging attack on watermarked video ( $\beta = 5$ ) with window size  $\lambda = 3$ . Each box indicates a separate frame of the watermarked video, with the watermarks embedded in each frame indicated.

watermarks in future frames. In the last frame of the set, the expected correlation is  $E[\rho(t)] = \frac{1}{2}\rho^*$ . In contrast with the  $\beta = 1$  case, a detection miss in the last frame is not fatal because the next state is already in the detector's queue. Making the simple assumption that the detector misses half the time on the last frame, the expected detection rate is then

$$\text{Detection Rate} = \frac{[(\beta - 1) + 0.5]}{\beta} = \frac{\beta - 0.5}{\beta} \quad (3.25)$$

For  $\beta = 3$ , the expected detection rate is approximately 83%. For  $\beta = 5$ , the expected detection rate is approximately 90%. And for  $\beta = 10$ , the expected detection rate is approximately 95%. The observed detection rates for the "Random Only" embedder in Figure 3.12 are in agreement with these expected rates.

Consider a window size of  $\lambda = 3$  and  $\beta = 3, 5, 10$ . Figure 3.14 shows a moving averaging window on the watermarked video for  $\beta = 5$ . There are three possible configurations for each window. Case I: For every set of  $\beta$  frames, the averaging window will contain three frames watermarked identically for the first  $\beta - 2$  frames. This averaging does not decrease the expected correlation when the averaged frame  $\hat{Y}(t)$  is examined, and  $E[\rho(t)] = \rho^*$  and watermark detection is very likely to succeed. Successful detection also provides the next state (shown as  $W_B$  in Figure 3.14) into the detector's queue. Case II: Near the end of the set, the window contains two

frames watermarked with  $W_A$  and a single frame watermarked with  $W_B$ . When  $\hat{Y}(t)$  is provided to the detector, the correlation will be

$$E[\rho(t)] = \frac{\gamma}{3N} [E[W_A \cdot W_K] + E[W_A \cdot W_K] + E[W_B \cdot W_K]] \quad (3.26)$$

When the detector correlates with the watermark  $W_K = W_A$ , then the expected correlation value is  $E[\rho(t)] = \frac{2}{3}\rho^*$ . This is significantly greater than the threshold of (3.17), so the detector is unlikely to miss. A similar result holds for the last configuration, where the averaging window contains two frames watermarked with  $W_B$  and a single frame watermarked with  $W_A$ . The detector will already have  $W_B$  in its queue, and correlating  $\hat{Y}(t)$  with  $W_K = W_B$  results in  $E[\rho(t)] = \frac{2}{3}\rho^*$ . Thus, for  $\lambda = 3$ , the smallest correlation value obtained from an attacked frame is  $E[\rho(t)] = \frac{2}{3}\rho^*$  and the detector rarely misses. This is why the detection rates for “Random Only” are 100% for  $\lambda = 3$  in Figure 3.12.

When the averaging window is  $\lambda = 4$  frames in size, a similar analysis reveals that for every  $\beta$  frames, there is one case where the attacked frame  $\hat{Y}(t)$  is constructed by averaging two frames with  $W_A$  and two frames with  $W_B$ , which then results in a correlation of  $E[\rho(t)] = \frac{1}{2}\rho^*$  when  $W_K = W_A$  or  $W_B$ . Thus, (3.25) also describes the expected detection rate for  $\lambda = 4$ .

### 3.6.2 Adaptive state transitions

The use of adaptive state transitions changes the temporal structure of the watermark based on the characteristics of the video, such that a minimum amount of temporal redundancy is present with respect to changing feature values. To compare the structure of the watermarked video when adaptive state transitions are used (“Adaptive”) and when adaptive state transitions are not used (“Features Only”), one can examine runs in the watermarked video. A run of length  $k$  is a set of  $k$  consecutive frames that have been watermarked by the same key, where the frame immediately before and after the run are watermarked using a different key. If adaptive state transitions are not used, the watermark is constructed such that all runs of

Table 3.1  
Structure of watermarked video using adaptive state transitions for *Akiyo*

Redundancy	Unique Keys	Number of Runs	Mean Run Length
$\alpha = 300 \quad \beta = 3$	98	98	3.06
$\alpha = 300 \quad \beta = 5$	56	56	5.36
$\alpha = 300 \quad \beta = 10$	26	26	11.54
$\alpha = 150 \quad \beta = 3$	98	98	3.05
$\alpha = 150 \quad \beta = 5$	56	56	5.36
$\alpha = 150 \quad \beta = 10$	27	27	11.11
$\alpha = 30 \quad \beta = 3$	90	99	3.03
$\alpha = 30 \quad \beta = 5$	49	57	5.25
$\alpha = 30 \quad \beta = 10$	20	28	10.71

the video are of length  $\beta$  frames. The average run length is thus  $\beta$ . When adaptive state transitions are used, run lengths vary across the video depending on when the feature values change. Run lengths depend on the video and the features.

For the “Adaptive” embedder, Tables 3.1–3.4 show the average run lengths of the watermarked video for the *Akiyo*, *Foreman*, and *Bus* sequences, as well as averages over all the videos. The number of unique keys used by the embedder to watermark the video are also noted. A key is unique if the same key has not been used to watermark an earlier frame of the video. The *Bus* video was only 150 frames in length so no results are available for  $\alpha = 300$ .

The *Akiyo* sequence consists of a woman speaking in front of a static background and has very little activity or motion. The *Foreman* sequence has more motion in the scene while the *Bus* sequence introduces a panning and zooming camera, which results in significant amounts of activity. Comparing the run lengths, the *Akiyo* sequence has run lengths that are very close to  $\beta$ . Short run lengths indicate that feature values are changing infrequently, such that the embedder is easily able to

Table 3.2  
Structure of watermarked video using adaptive state transitions for *Foreman*

Redundancy	Unique Keys	Number of Runs	Mean Run Length
$\alpha = 300 \quad \beta = 3$	65	65	4.61
$\alpha = 300 \quad \beta = 5$	29	29	10.34
$\alpha = 300 \quad \beta = 10$	9	9	33.33
$\alpha = 150 \quad \beta = 3$	65	65	4.62
$\alpha = 150 \quad \beta = 5$	30	30	10.00
$\alpha = 150 \quad \beta = 10$	10	10	30.00
$\alpha = 30 \quad \beta = 3$	62	68	4.41
$\alpha = 30 \quad \beta = 5$	30	66	8.43
$\alpha = 30 \quad \beta = 10$	11	16	18.70

Table 3.3  
Structure of watermarked video using adaptive state transitions for *Bus*

Redundancy	Unique Keys	Number of Runs	Mean Run Length
$\alpha = 300 \quad \beta = 3$	—	—	—
$\alpha = 300 \quad \beta = 5$	—	—	—
$\alpha = 300 \quad \beta = 10$	—	—	—
$\alpha = 150 \quad \beta = 3$	29	29	5.17
$\alpha = 150 \quad \beta = 5$	8	8	18.75
$\alpha = 150 \quad \beta = 10$	2	2	75.00
$\alpha = 30 \quad \beta = 3$	28	30	5.00
$\alpha = 30 \quad \beta = 5$	10	12	13.07
$\alpha = 30 \quad \beta = 10$	3	4	38.25

Table 3.4  
 Structure of watermarked video using adaptive state transitions  
 (Mean over all videos)

<b>Redundancy</b>	<b>Unique Keys</b>	<b>Number of Runs</b>	<b>Mean Run Length</b>
$\alpha = 300 \quad \beta = 3$	82	82	3.84
$\alpha = 300 \quad \beta = 5$	43	43	7.85
$\alpha = 300 \quad \beta = 10$	18	18	22.44
$\alpha = 150 \quad \beta = 3$	64	64	4.28
$\alpha = 150 \quad \beta = 5$	31	31	11.37
$\alpha = 150 \quad \beta = 10$	13	13	38.70
$\alpha = 30 \quad \beta = 3$	60	66	4.15
$\alpha = 30 \quad \beta = 5$	30	45	8.92
$\alpha = 30 \quad \beta = 10$	11	16	22.55

find  $\beta$  consecutive frames with constant features. In fact, for *Akiyo*, the run lengths are very close to  $\beta$ . On the other hand, the run lengths increase as the amount of motion and activity in the video increases in the *Foreman* and *Bus* sequences, which indicate rapidly changing feature values. When feature values change rapidly, the embedder does not observe  $\beta$  frames with constant features. For example, finding  $\beta = 10$  consecutive frames with constant features within *Bus* is very difficult and the run lengths far exceed  $\beta$ . (This watermark is nearly time-invariant.)

### 3.7 Conclusions and Future Work

Temporal synchronization was examined using a new framework for blind symmetric video watermark embedding and detection. The embedder and detector models are general and encompass the behavior of most current video watermarking techniques, including those techniques which produce time-invariant key, time-independent key, and time-periodic key watermarks. The framework demonstrates the relationship between temporal redundancy and temporal synchronization.

- A watermark without temporal redundancy is vulnerable to frame dropping and transposition attack (Theorem 3.4.3). The  $\phi(\cdot)$  function induces a “chain of states” that is created by the embedder (through the state transitions in the key generator) and traced by the detector (using the queue and state predictor). Frame dropping and transposition severs this chain, causing the detector to lose synchronization. This vulnerability is particularly a concern for time-independent key watermarks.
- By designing a watermark with temporal redundancy, an arbitrary amount of robustness can be achieved against frame dropping, decimation, and transposition. In particular, a watermark can be designed (such as the watermark discussed in Section 3.5) to be robust unless  $\beta$  consecutive frames are dropped (Theorem 3.5.1), but at a tradeoff in security.

- Insertion of an arbitrary number of unwatermarked frames will not cause the detector to lose synchronization, because unwatermarked frames do not affect the detector’s queue (Theorem 3.4.2). Temporal upsampling will not desynchronize the detector (Lemma 8).
- Initial synchronization is trivial for time-invariant key watermarks because every frame is watermarked with a key in  $\mathcal{S}_0$ , but is only at the beginning of the video for a time-independent key watermark, where keys do not repeat. Initial synchronization may occur at resynchronization frames (every  $\alpha$  frames) in the watermark described in Section 3.5. A watermark with many resynchronization frames is vulnerable to attacks which attempt to identify and remove these frames. Some methods for improving the security of resynchronization frames were mentioned in Section 3.5.2.

One of the contributions of the framework is the modeling of the key generator using state machines. This opens avenues for further research. The graph structure of the state machine, induced by representing states as vertices and  $\phi(\cdot)$  as edges, affects temporal redundancy. For example, a key generator using a state machine with a strongly connected graph representation may be more robust or secure. Instead of watermarking multiple frames with the same key, temporal redundancy may be inserted into the key schedule by altering or exploiting the structure of the graph. This observation leads to the desire to quantify temporal redundancy in a more general and formal manner than  $\alpha$  and  $\beta$ . This is challenging, as the framework shows that temporal redundancy is useful for both local sets of frames ( $\beta$ ) as well as globally throughout the watermark ( $\alpha$ ). Quantifying temporal redundancy by simply one measure, such as redundancy in the information-theoretic sense [132], is not necessarily insightful. An extended analysis may involve (Hidden) Markov Models [295], Kalman filtering, or other state-estimation techniques.

Related to quantifying temporal redundancy is the tradeoff in security. The discussion in Section 3.1, as well as estimation and collusion mentioned in other

works [82, 89], support the claim that increased redundancy reduces security. However, demonstrating a concrete relationship between redundancy and security is a goal for future work. In essence, we would like to consider how difficult an arbitrary watermark signal is to estimate using collusion and other attacks. For example, one could try to estimate watermarks with varying redundancy by using various signal estimators (such as simple arithmetic mean, Weiner estimation) as a preliminary step to a more in-depth examination of security issues.

Another avenue for exploration is the prediction mechanism. One extension is to extend the detector state prediction from predicting the next state ( $\tilde{s}(t+1)$ ) to predicting multiple states in time ( $\tilde{s}(t+1), \tilde{s}(t+2), \dots$ ). Another extension is to provide the state predictor additional side-information (such as by a template) to improve predictions.

A method for obtaining a measure for the “goodness” of the feature extraction is desired. There are many functions that may be used for obtaining features, such as the simple features used in these experiments and more sophisticated functions such as a “visual hash” [296]. Because different applications require different degrees of robustness and expect different attacks on the watermark, the feature set is likely to be application dependent. Amongst the criteria for good features include (1) the degree of robustness of the feature against attacks, (2) the number of distinct feature values that can be produced for any video frame, (3) the degree of sensitivity of the feature to the characteristics of a video frame, and (4) computational costs. We would like to determine properties of functions which are most helpful for improving the robustness and security of the watermark.

The embedder model uses feature extraction for each frame to affect the watermark structure of future frames. This can be compared, in terms of robustness, security, and invisibility, to frame-dependent techniques, such as [89], where features of the video frame are used to affect the watermark structure of the current frame.

The adaptive state transition model described in Section 3.5.1 simple, but has the shortcoming that state transitions occur when the video is relatively static. This can



introduce flicker when the watermarked video is viewed. Flicker was not observed in the experiments because the power of the embedded watermark was sufficiently small, but flicker may be an issue when the embedding power is increased. A more sophisticated adaptive model would also consider the video before deciding a state transition. For example, the state transition may be deferred until the *next* frame when feature values change, after the feature values have been constant for at least  $\beta$  consecutive frames.

The examination of synchronization in watermark detection has often focused on designing templates, either by embedding an explicit synchronization signal, organizing the watermark to produce such a signal, or by using salient features of the video for synchronization. However, there has been relatively little work in modeling the watermark embedding and detection processes for synchronization similarly to the approach taken in this chapter. Such models can be used to show why some watermarks (such as time-invariant key watermarks for temporal synchronization and tiled watermarks for spatial synchronization) are “easier” to synchronize than others, even when the watermark embedding does not take advantage of any special transform or invariance properties, or the insertion of an explicit synchronization signal.

## 4. SPATIAL SYNCHRONIZATION

In the previous chapter, a framework was proposed for temporal synchronization in video watermarking. In the framework, the watermark embedder creates the watermark embedded into each frame of the video using a state machine key generator. The objective of the watermark detector is to determine, for each frame of the input video, which key was used to produce the watermark embedded in that frame. The key generators that produce time-invariant key, time-periodic key, and time-independent key schedules are special cases. A technique for efficient temporal synchronization was described by designing temporal redundancy in the watermark key sequence. Only temporal synchronization was addressed.

In this chapter, the temporal synchronization framework is adapted and extended to spatial synchronization of still image and video watermarks. Like temporal synchronization, redundancy in the watermark structure can be used to produce a template for efficient spatial synchronization. Specifically, the watermark embedder constructs a watermark with a regular or “tiled” structure. When the autocorrelation of the watermarked image is obtained, peaks (local maxima) occur in a pattern resembling a grid, and this pattern is examined to estimate the rotation and the scale of the watermark. The redundant structure of a tiled watermark reduces security because such a watermark is more easily estimated.

Previous works have considered synchronization by using the autocorrelation or constructing a tiled watermark structure. Kutter [267] constructs the watermark by the superposition of an elementary watermark signal and shifted versions of the same signal. The shifted watermarks induce peaks in the autocorrelation, which serves as the basis for obtaining the watermark rotation and scale. Kalker [38] constructs a tiled watermark for efficient synchronization under spatial translation. Alattar [260]

constructs a tiled watermark, and then uses a log-polar mapping of the autocorrelation to estimate the watermark rotation and scale. Deguillaume [266] also constructs a tiled watermark, but the affine coordinate transformation is estimated by using the Hough Transform. Unfortunately, the Hough Transform is computationally expensive to obtain, particularly if the rotation and scale is estimated to a high degree of precision.

The synchronization method described in this chapter is similar to those proposed by Alattar [260] and Deguillaume [266] in that the grid-like pattern of peaks in the autocorrelation is used to estimate the position of the watermark. However, an alternate method for watermark scale and rotation estimation is proposed, other than the log-polar mapping and Hough Transform. This method is limited to uniform scaling and rotation attacks, which is more restrictive than [266] and identical to the restrictions of [260]. However, the method described here does not require the detector to obtain the Hough Transform of the autocorrelation. In addition, a method for constructing watermarks with varying the degree of spatial redundancy is described here, while [260, 266] focuses on the template matching while leaving watermark construction open.

#### 4.1 Watermark Embedding

A straightforward approach for extending the temporal synchronization framework to spatial synchronization is to construct the watermark in a block-by-block basis. The watermark embedder is shown in Figure 4.1. The original image and the embedding key  $K_E$  are provided as inputs. Let  $X(n)$  be a block of the original image, where  $n = 0, 1, \dots$  is the block index. The key generator provides the key  $K(n)$  to the watermark generator, which uses the key to produce the watermark signal  $W(n)$ .  $W(n)$  is inserted into  $X(n)$  to produce the watermarked block  $Y(n)$ . The block analyzer examines  $Y(n)$  to produce a feature vector  $F(n)$ , which is used by the key generator to produce the key to watermark the next block  $X(n+1)$ . This

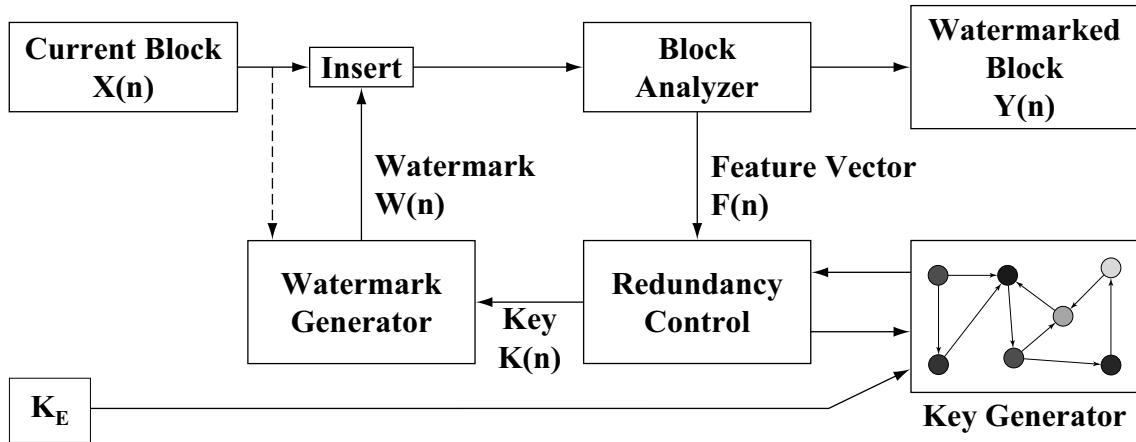


Fig. 4.1. Watermark embedder with state machine key generator for spatial synchronization

process is repeated until all blocks of the image have been watermarked. The redundancy control “resets” the key generator at regular intervals to control the structure of the watermark, which will be described below.

The key generator is a state machine whose internal state is denoted by  $s(n) \in \mathcal{S}$ , where  $\mathcal{S}$  is the set of states. When block  $X(n)$  is watermarked, the key generator supplies key  $K(n) = \lambda(s(n))$  to the watermark generator. When the watermarked block  $Y(n)$  is produced and the feature vector  $F(n)$  is available, the key generator performs a state transition:  $s(n+1) = \phi(s(n), K_E, F(n))$ , where  $\phi(\cdot)$  is the state transition function. A cryptographic hash function, where  $s(n)$ ,  $K_E$  and  $F(n)$  are hashed, is an example state transition function (equations (3.6) and (3.7).) Prior to watermarking the first block of the image, the state of the key generator is set to an initial state  $s_0 \in \mathcal{S}$ , which may be dependent on  $K_E$ <sup>1</sup>. The key used to watermark the first block is thus  $K(0) = \lambda(s_0) = K_0$ . Also, when the key generator is reset by the redundancy control, its internal state is set to  $s(n) = s_0$  and the key used to watermark the block is  $K(n) = K_0$ .

<sup>1</sup>The key generator may have randomized (non-deterministic) state transitions as in chapter 3. However, in this chapter, the state machine must have a unique initial state.

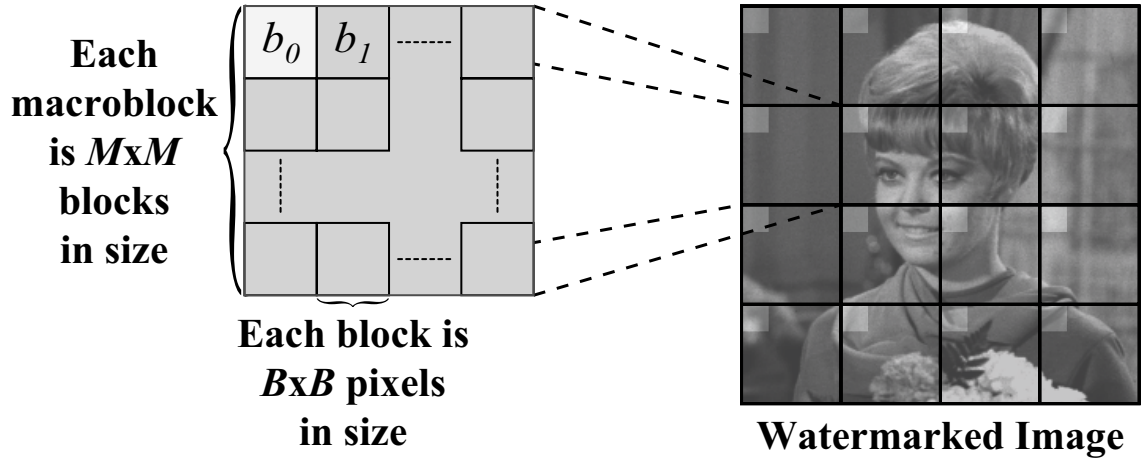


Fig. 4.2. Watermark structure showing macroblocks and blocks. The image shows an example of the structure of a watermark constructed and inserted in the spatial domain. The first block of all macroblocks are identical.

This scheme can produce a watermark with a structure resembling that shown in Figure 4.2, where the watermark is partitioned into non-overlapping macroblocks  $\Gamma_0, \Gamma_1, \dots$ . Each macroblock is a region of  $M \times M$  blocks, with blocks  $b_0, b_1, \dots, b_{M^2-1}$ . Each block is a region of  $B \times B$  pixels. Assume, for convenience, that the dimensions of the image are integral multiples of  $MB$ . To produce the watermark structure, the key generator is reset prior to watermarking the first block ( $b_0$ ) of each macroblock. This block is watermarked by using the key  $K_0 = \lambda(s_0)$ . After this block is watermarked, the key generator performs state transitions to construct the watermark for the remaining blocks  $b_1, \dots, b_{M^2-1}$ . After the last block of the macroblock has been watermarked, the key generator is reset by the redundancy control to watermark the first block of the next macroblock. There are other watermark structures that can be produced by this framework (such as by changing the redundancy control or the order which the blocks of the image are watermarked), however only this watermark structure will be considered in this chapter.

The first block ( $b_0$ ) of all the macroblocks are generated by the key  $K_0$ . If  $W(n)$  is only a function of  $K(n)$ , then these blocks are identical.<sup>2</sup> These synchronization blocks form a regular pattern, or spatial redundancy, which shall be the basis for the spatial synchronization technique discussed in Section 4.2. The remaining blocks of the watermark, the non-synchronization blocks, are generated by keys that depend on the block analyzer and are generally not identical. Thus, only the synchronization blocks are redundant. Although the non-synchronization blocks do not play a role in synchronization, they are part of the watermark and can be detected by the watermark detector. For example, the non-synchronization blocks may encode a payload.

The degree of spatial redundancy in the watermark can be adjusted by changing  $M$  and  $B$ . For  $M = 1$ , the watermark is constructed by regularly repeating, or tiling, an elementary watermark signal of size  $B \times B$  pixels. All the blocks are synchronization blocks and there are no non-synchronization blocks. For  $M > 1$ , spatial redundancy is decreased because the watermark contains non-synchronization blocks. As  $M$  increases, the fraction of synchronization blocks to non-synchronization blocks decreases. Increasing  $B$  does not directly reduce the ratio between synchronization blocks to non-synchronization blocks, but any increase in the product  $MB$  decreases the number of macroblocks that can “fit” the area of an image of fixed size. Because each macroblock contains a single synchronization block, decreasing the number of macroblocks of the watermark decreases the total number of synchronization blocks, and hence, spatial redundancy of the watermark.

## 4.2 Watermark Synchronization

When the watermarked image is provided to the watermark detector, the detector establishes synchronization and then detects the watermark. In this section, the

---

<sup>2</sup>In particular, it is assumed that the watermark generator does not change or adapt the watermark structure of the synchronization blocks based on the original image block, except for possibly amplitude scaling (perceptual shaping).

focus shall be on the synchronization process, when the watermarked image may be attacked by uniform scaling and rotation. The objective of the synchronizer is to estimate the rotation and scale of the embedded watermark. Once the synchronizer estimates the rotation and scale, these transformations are “reversed” to obtain a normalized watermark. Detecting the normalized watermark is a straightforward adaptation of the Watermark Detection Protocol (see Section 3.3) from frame-by-frame detection to block-by-block detection, and will not be described here. Also, this synchronization process does not estimate the translation (spatial shifting) of the watermark. The shift be estimated in a manner similar to Kalker [38] or by using an auxiliary template, or even by blind search. Because watermark detection can occur at any macroblock, the translation search is only needed up to the size of a single macroblock, not the size of an image.

For a watermark with the structure described in Section 4.1, the synchronization blocks are repeated at regular intervals. When the autocorrelation of the watermark is obtained, the synchronization blocks induce local maxima, or peaks, with a grid-like arrangement similar to Figure 4.3(a). The distance between two adjacent peaks of the grid is the neighbor distance  $\chi$ . If the watermark has not been attacked,  $\chi = BM$  pixels. Uniformly scaling the watermark causes the distance of the peaks to change to  $\chi = BMf$  pixels, where  $f$  is the (unknown) scaling factor. The scaling factor is the ratio of each dimension (height, width) of the attacked image to the corresponding dimension of the watermarked image. Figure 4.3(b) illustrates the effect of uniform scaling on the autocorrelation peaks of the watermark. Rotating the watermark does not change the neighbor distance, but causes the peaks to be rotated by the same rotation angle  $\theta$  as shown in Figure 4.3(c). Thus, if the watermark has been re-scaled by unknown scaling factor  $f$  and rotated by unknown angle  $\theta$ , the synchronizer can estimate the watermark scale and rotation by estimating  $\chi$  and  $\theta$ .

Several issues complicate the synchronization process. First, the signal power of the embedded watermark is much lower than that of the original image. The autocorrelation of the watermarked image may not show the desired grid of peaks unless

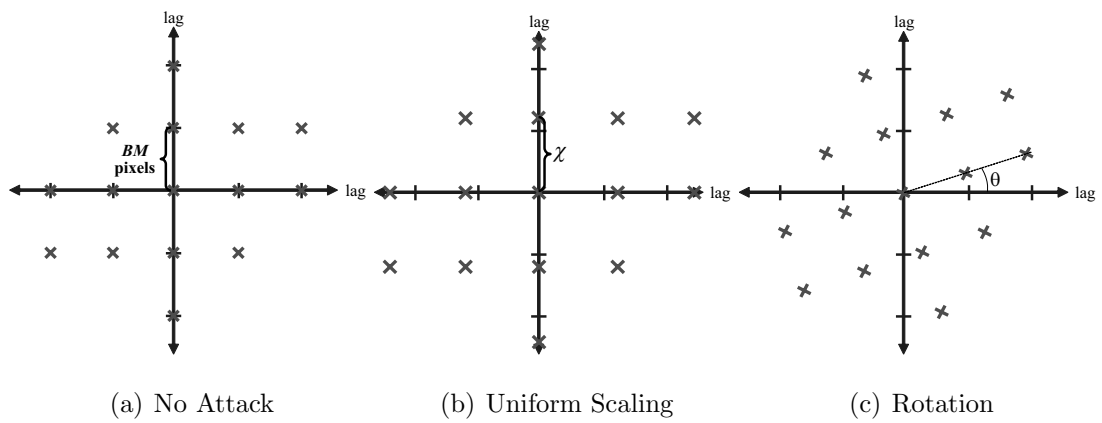


Fig. 4.3. Autocorrelation of watermark showing location of local maxima or peaks. Peaks are indicated by  $\times$ , forming a regular grid. Not all the peaks in the autocorrelation are shown.



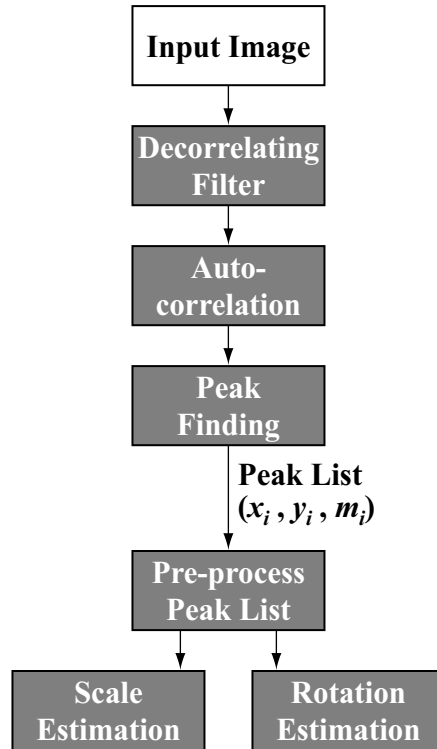


Fig. 4.4. Spatial synchronization procedure

the host-signal interference is reduced prior to obtaining the autocorrelation. Another issue is that the autocorrelation of a watermarked image may have additional peaks, missing peaks, and peaks that are slightly moved (or “perturbed”) from their ideal positions.

The synchronization process is shown in Figure 4.4. The decorrelating filter (3.8) is convolved with the input image to reduce host-signal interference, and then the autocorrelation of the filtered image is obtained. Then, the location and magnitude of all local maxima of the autocorrelation is obtained. The method for peak finding is simple: A window is moved across the autocorrelation and the local maximum inside the window is obtained. If the center value is the maximum, then it is a peak. After peak finding, a list of peaks  $\mathcal{P} = \{p_i\}$ ,  $p_i = (x_i, y_i, m_i)$  is obtained, where  $x_i$  and  $y_i$  are the coordinates (position) of peak  $p_i$  in the autocorrelation and  $m_i$  is the magnitude of  $p_i$ .

Since the autocorrelation function of an image is symmetric about the origin, half the peaks are redundant and are removed from the peak list. (As an optimization, the autocorrelation would only need to be obtained from lags over a half-plane, instead of obtaining the (full) autocorrelation and then removing half of the peaks.) Then, the peak list is truncated to retain only the  $P$  peaks of greatest magnitude. Retaining too few peaks does not allow the grid structure of Figure 4.3 to be detected, and retaining too many peaks causes an excessive number of “noise” peaks (or peaks that are not part of the grid structure). The value  $P = 50$  (chosen empirically) was used in the experiments, which provided a reasonable balance between detecting too many noisy peaks and too few peaks.

The image shown in Figure 4.5 will be used as an example to illustrate the scale and rotation estimation. This image has been watermarked in the spatial domain ( $M = 3$ ,  $B = 16$ ) and attacked by scaling the image using  $f = 1.15$  followed by six degrees rotation. Figure 4.5(a) shows the watermarked and attacked image. The positions of the peaks in  $\mathcal{P}$  are shown in Figure 4.5(b). The grid pattern is present, but also accompanied by a number of additional “noise” peaks that arise from the interference of the original image. The neighbor distance is  $\chi = MBf = 55.2$  pixels.

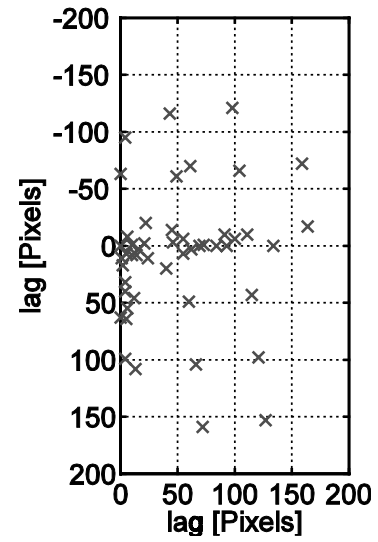
#### 4.2.1 Watermark scale estimation

The peaks of  $\mathcal{P}$  are assumed to be spatially arranged in a grid with unknown neighbor distance  $\chi$  and oriented with unknown angle  $\theta$ . The objective of the scale estimation is to estimate  $\chi$ . The scale estimation technique considers the distances between all pairs of peaks, for two reasons. First, the distance between any two peaks of the grid is independent of the orientation (or rotation) of the grid. Second, the arrangement of the peaks implies that certain distances should occur frequently, which becomes the basis by which the watermark scale can be estimated. Thus, the first step is to obtain the distance

$$d_{ij} = d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (4.1)$$



(a) Attacked image



(b) Peak positions

Fig. 4.5. Example of watermark scale and rotation estimation using image *Girl*. PSNR of watermarked image is 33.8 dB,  $M = 3$ ,  $B = 16$ . Attack is re-scaling by factor  $f = 1.15$  followed by rotation of  $6^\circ$ . Neighbor distance is  $\chi = MBf = 55.2$  pixels.

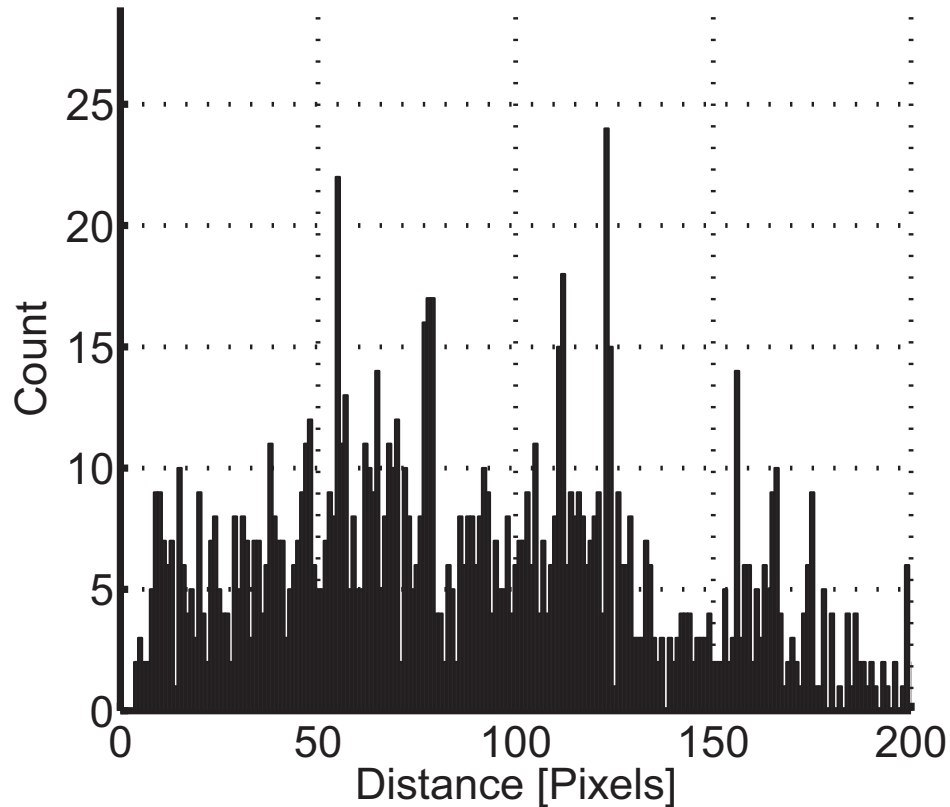


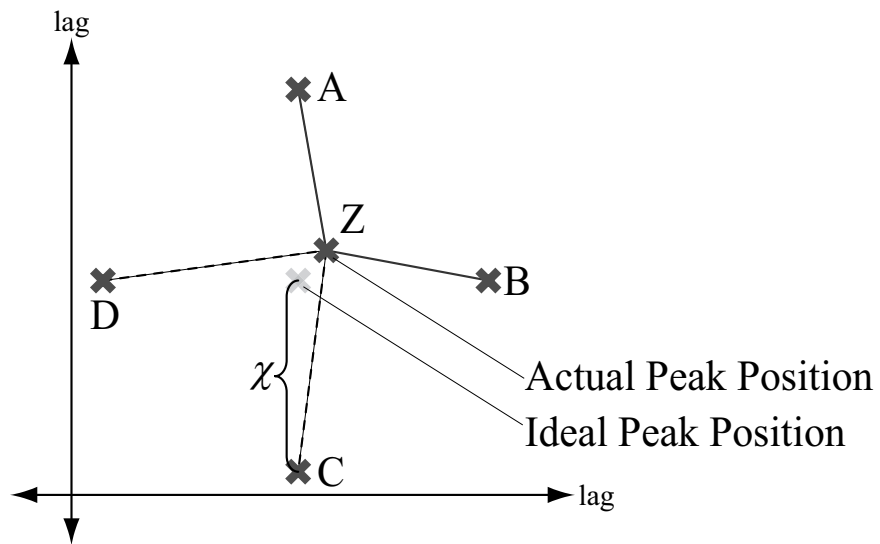
Fig. 4.6. Distance histogram of *Girl* (histogram bin size  $\Delta_s = 1$ ). Large values occur at the bins corresponding to the distances of 55 pixels, 78 pixels, 110 pixels, and 123 pixels.

between every pair of peaks  $p_i, p_j \in \mathcal{P}$ ,  $i \neq j$ . There are  $P$  peaks in  $\mathcal{P}$ , so there are  $\frac{1}{2}P(P - 1)$  pairs of peaks. The pairwise peak distances are used to construct a distance histogram  $\mathcal{D}$ . The value of each bin in the histogram is the number of  $d_{ij}$ 's in the interval  $[(n - \frac{1}{2})\Delta_s, (n + \frac{1}{2})\Delta_s)$  for  $n = 0, 1, 2, \dots$  and histogram bin size  $\Delta_s > 0$ . The bins with high values are important because these bins correspond to  $d_{ij}$ 's which occur most frequently. “Noise” peaks are not likely to form a regular pattern that gives rise to high bin values. The distance histogram of *Girl* is shown in Figure 4.6. The bin corresponding to the actual neighbor distance of 55.2 pixels has a high value, however it is not the bin with the maximum value in the histogram.

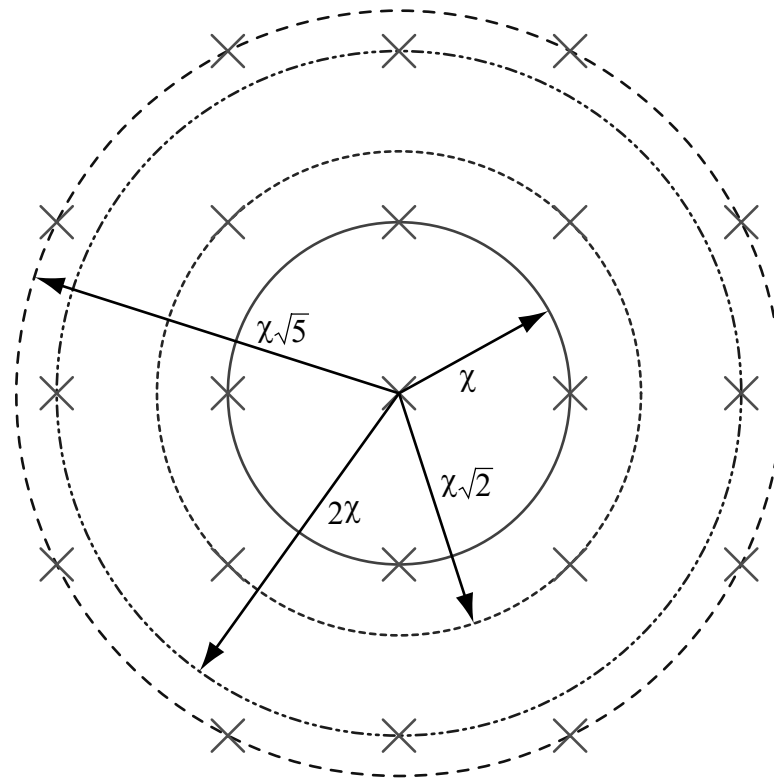
The construction of the distance histogram from the pairwise peak distances does not consider the fact that the peaks may be slightly shifted or “perturbed” from their ideal positions. Small shifts in the peak positions may arise when the image pixel values are interpolated after the coordinate transformation. Interpolation has the effect of “blurring” the image, which can cause the peaks in the autocorrelation to not be as “sharp” or “well-defined” as the non-interpolated image. The peak finder may also have limited precision, such that peaks can only be identified with single pixel (as opposed to sub-pixel) resolution. Lastly, nearby peaks arising from the original image may also confuse the peak finder, leading to a “perturbed” peak. Figure 4.7(a) illustrates the effect when a peak is slightly shifted from its ideal position. In the ideal case where there is no perturbation in the positions of the peaks, each of the distances between the non-center peaks (A,B,C,D) to the center peak (Z) is  $\chi$ . When  $\mathcal{D}$  is obtained, there is a high value at the bin whose distance interval contains  $\chi$ . However, when the peaks are not in their ideal locations then some pairwise distances become slightly longer and other distances become slightly shorter. These distances may map onto different bins in  $\mathcal{D}$ . When this occurs, the value of the bin whose distance interval contains  $\chi$  is reduced and estimating  $\chi$  from  $\mathcal{D}$  is more challenging.

The effect of perturbed peaks may be reduced by smoothing the distance histogram. Specifically, the distance histogram  $\mathcal{D}$  is treated like a discrete one-dimensional signal  $\mathcal{D}(n)$  where the bin values are the values of the signal, and smoothing is accomplished by convolving  $\mathcal{D}$  with a smoothing filter  $h_s$  to generate the processed distance histogram  $\mathcal{D}^* = \mathcal{D} * h_s$ . The distance intervals of the bins of  $\mathcal{D}^*$  correspond to the same intervals as the bins of  $\mathcal{D}$ . Smoothing the distance histogram allows the value of each histogram bin to influence the value of nearby bins, in recognition that the pairwise peak distances may be affected by a small amount of noise.

While the distance histogram provides some information regarding  $\chi$ , the estimation of  $\chi$  can be improved by using additional geometric properties of the grid. Figure 4.7(b) shows a regular grid of peaks with neighbor distance  $\chi$ . The center



(a) Effect of shifted or "perturbed" peak on pairwise peak distances



(b) Geometric structure of grid peaks

Fig. 4.7. Improving the scale estimation

peak represents any peak on the grid. The four nearest neighboring peaks lie at the distance  $\chi$  from the center peak. Estimating  $\chi$  directly from  $\mathcal{D}$  or  $\mathcal{D}^*$  uses only these four peaks to infer a grid structure. However, a regular grid structure also has four peaks at a distance of  $\chi\sqrt{2}$  from the center peak, four additional peaks at a distance  $2\chi$ , and eight peaks at the distance of  $\chi\sqrt{5}$  from the center peak. Observing all these distances suggests the presence of a grid more than merely observing the distance  $\chi$ . The distance histogram of *Girl* (Figure 4.6, with  $\chi = 55.2$  pixels) shows large values at bins whose distance intervals include the distances 55.2 pixels,  $55.2\sqrt{2} \approx 78.0$  pixels, 110.4 pixels, and  $55.2\sqrt{5} \approx 123.4$  pixels.

A score function is used to estimate the neighbor distance. The score function  $S_s$  is a discrete function with values at  $k\Delta_s$  for integral  $k$ :

$$S_s(k\Delta_s) = q_s(k\Delta_s) + q_s(\sqrt{2}k\Delta_s) + q_s(2k\Delta_s) + q_s(\sqrt{5}k\Delta_s) \quad \text{for } k = 1, 2, \dots \quad (4.2)$$

where  $q_s(z)$  is the value of the bin in  $\mathcal{D}^*$  whose distance interval contains  $z$ . For fixed  $k$ , the score  $S_s(k\Delta_s)$  is a quantity that reflects the likelihood (based on the pairwise distances) that the autocorrelation peaks form a grid with neighbor distance  $\hat{\chi} = k\Delta_s$ . Having obtained  $S_s(\cdot)$ , the watermark detector detects the watermark using the neighbor distance which has the largest score ( $\hat{\chi} = \arg \max S_s(k\Delta_s)$ ). If the watermark is not detected, then the watermark detector attempts to detect the watermark using the neighbor distance with the second highest score. This process is repeated in decreasing order of score until the watermark is detected or the watermark detector “gives up.”

The score function obtained for the *Girl* image is shown in Figure 4.8. The maximum score occurs at the distance of  $\hat{\chi} = 55$  pixels so the detector successfully estimates the scale of the watermark on the first attempt. If the first estimate was not the correct neighbor distance, the detector would then try  $\hat{\chi}_2 = 56$  pixels, then  $\hat{\chi}_3 = 39$  pixels,  $\hat{\chi}_4 = 78$  pixels, and so on.

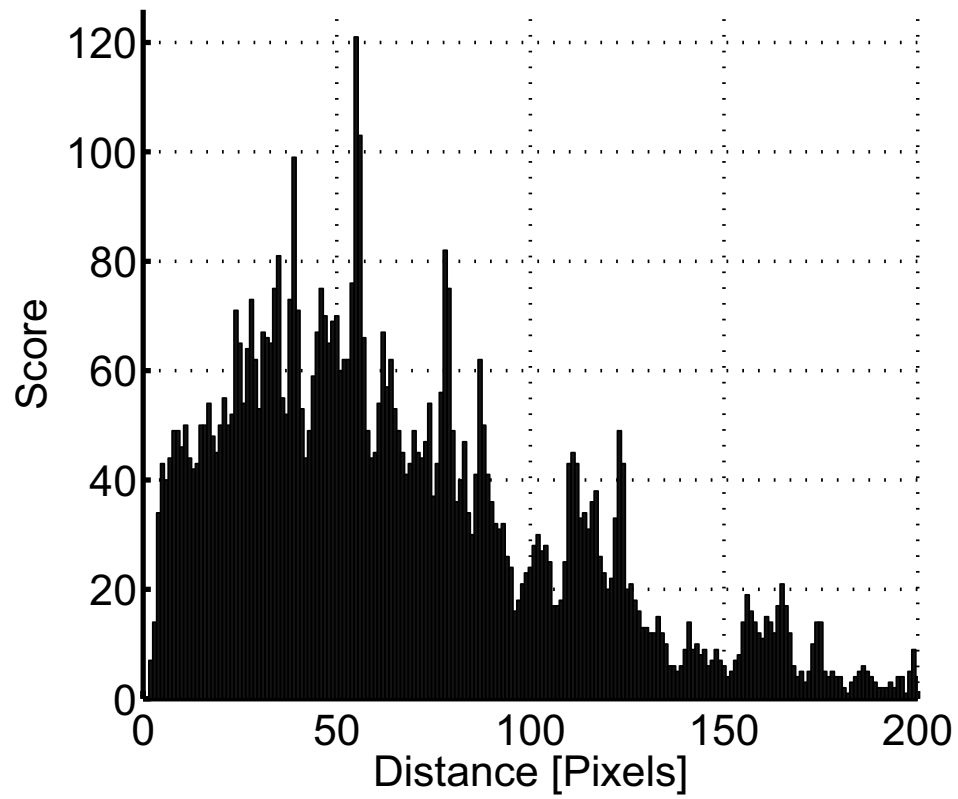


Fig. 4.8. Score function  $S_s(k\Delta_s)$  of *Girl* ( $\Delta_s = 1$ ). The largest score occurs for the distance of 55 pixels, which implies that scale estimation is correct on the first attempt.



### 4.2.2 Watermark rotation estimation

The estimation of the watermark orientation is challenging because of the symmetry of the template. Rotating the watermark by angles of  $\theta$ ,  $\theta + \frac{\pi}{2}$ ,  $\theta + \pi$ , and  $\theta + \frac{3\pi}{2}$  produce indistinguishable patterns of peaks. Horizontal and vertical flipping also produce indistinguishable patterns. Similar to [266], all eight ambiguities arising from combinations of rotation and flipping must be searched by the detector if other means for distinguishing these transformations are not used, such as another template.

The objective of watermark rotation estimation is to estimate the orientation of the grid of peaks  $\theta$ , where  $0 \leq \theta < \frac{\pi}{2}$ . The rotation by angles  $\theta + k\left(\frac{\pi}{2}\right)$ ,  $k = \dots, -1, 0, 1, \dots$ ,  $0 \leq \theta < \frac{\pi}{2}$  are considered congruent. The technique for rotation estimation is conceptually similar to the scale estimation. The pairwise angles

$$\psi_{ij} = \psi(p_i, p_j) = \arctan\left(\frac{y_i - y_j}{x_i - x_j}\right) \quad (4.3)$$

are used to construct the *angle histogram*  $\mathcal{A}$ . The bins of  $\mathcal{A}$  partition the interval  $[0, \frac{\pi}{2})$  to intervals of the histogram bin size  $\Delta_r > 0$ . For example,  $\Delta_r = 1^\circ = \frac{\pi}{180}$  uses 90 bins in  $\mathcal{A}$ .

Once the angle histogram is obtained, it is smoothed by convolving with the smoothing filter  $h_r$  to produce the smoothed histogram  $\mathcal{A}^* = h_r * \mathcal{A}$ . Unlike the smoothing for the distance histogram, angles are congruent in  $[0, \frac{\pi}{2})$  and the convolution is circular. After histogram smoothing, the rotation score is obtained

$$\begin{aligned} S_r(k\Delta_r) &= q_r(k\Delta_r) + q_r\left(k\Delta_r + \frac{\pi}{6}\right) + q_r\left(k\Delta_r + \frac{\pi}{4}\right) + q_r\left(k\Delta_r + \frac{\pi}{3}\right) \\ &\text{for } 0 \leq k\Delta_r < \frac{\pi}{2} \end{aligned} \quad (4.4)$$

where  $q_r(z)$  is the value of the bin of  $\mathcal{A}^*$  whose interval contains the angle congruent to  $z$  in  $0 \leq \theta < \frac{\pi}{2}$ . Similarly to estimating the neighbor distance, the watermark detector first estimates the rotation to be  $\hat{\theta} = \arg \max S_r(k\Delta_r)$ . If the watermark is not detected using  $\hat{\theta}$ , then the detector attempts the angle with the second highest

score for  $\hat{\theta}_2$ , and so on until either the watermark is detected or the detector gives up.

### 4.3 Experimental Results

To evaluate the synchronization technique, an additive watermark in the spatial domain was implemented. The watermark embedder embeds the watermark in the luminance of the original image block  $X(n)$  to produce the watermarked image block  $Y(n) = X(n) + \gamma W(n)$ , where  $W(n)$  is a zero mean, unit variance Gaussian signal produced by a pseudo random number generator seeded with  $K(n)$ , and  $n$  is the block index. The block analyzer ignores the watermarked image block and produces the feature  $F(n)$  by generating a random (32-bit) number. The key generator's state transition function hashes the current state, embedding key, and features to produce the next state, using SHA-1 [294] as the hash function with (3.6). Using the hash function and the random feature values effectively create random keys  $K(n)$  for all non-synchronization blocks, causing the corresponding  $W(n)$  to be uncorrelated. Correlation amongst the non-synchronization blocks may be beneficial to the synchronizer by strengthening some of the peaks in the grid of the autocorrelation, and is avoided in the experiments.

The original images were *Girl*, *Fruit*, *Crowd*, and *Peppers*, with each image  $256 \times 256$  pixels in size. The embedding strength was  $\gamma = 5.0$ , which produced watermarked images with PSNR of 33.8 dB compared with the original image. Perceptual shaping reduces the visibility of the watermark and may improve performance, however perceptual shaping is not used in these experiments to minimize the number of experimental parameters. The original and watermarked images are shown in Figures 4.9 and 4.10. Embedding parameters were macroblock sizes  $M = 1, 2, 3$  blocks and block sizes  $B = 8, 16, 32$  pixels. Synchronization attacks were performed as a transformation (either scaling or rotation) followed by bicubic interpolation to re-sample the attacked image. When scaling and rotation were both performed on a

single image, scaling was performed, followed by bicubic interpolation, followed by a rotation and a second bicubic interpolation.

The watermark synchronization uses a peak list of  $P = 50$  peaks. For scale estimation,  $\Delta_s = 1$  pixel, which allows  $\chi$  to be estimated to the nearest pixel. For histogram smoothing, the FIR filter kernel  $h_s$  is  $[0.15 \ 0.50 \ 1.00 \ 0.50 \ 0.15]$  where 1.0 is the value of  $h_s(0)$ . The design of this filter was ad hoc. A more rigorous design of  $h_s$  would assume a probability model for the peak perturbations and then use the probability model to determine the filter, which may be dependent on  $\Delta_s$ . For rotation estimation,  $\Delta_r = 1^\circ$ , which allows  $\theta$  to be estimated to the nearest degree. Histogram smoothing was not performed for rotation estimation so  $\mathcal{A}^* = \mathcal{A}$ . In these experiments, it is *assumed* that  $\Delta_s = 1$  pixel and  $\Delta_r = 1^\circ$  offers sufficient precision necessary for successful watermark detection, although the necessary precision is dependent on the watermarking technique in practice. Additional search may also be performed to refine the scale and rotation estimation.

#### 4.3.1 Scale and rotation rank

The synchronization performance is measured in terms of the number of attempts that the watermark detector requires before obtaining the correct scale and orientation. In Section 4.2.1, it was described that the watermark detector estimates the neighbor distance of the grid (which estimates of the watermark scale) by choosing the distance  $d$  with the highest score  $S_s(d)$ . If this estimated scale is not correct, the watermark detector estimates the scale using the distance with the second highest score. Consecutive attempts continue in decreasing order of  $S_s(\cdot)$  until the watermark detector correctly estimates the watermark scale. The scale rank is defined as the total number of attempts that the synchronizer requires to obtain the correct watermark scale, to the precision of  $\Delta_s$ . For example, if the detector correctly estimates the scale of the watermark on the first attempt, the scale rank is 1, which is the best rank. A large scale rank indicates a failure in the scale estimation technique.

For the purposes of obtaining the scale rank, the number of attempts (or searches) by the synchronizer is not constrained. A hypothetically flawed synchronizer which never searches the proper scale given unlimited searches would have a scale rank of  $\infty$ . The rotation rank is similarly defined as the total number of attempts that the watermark detector requires to obtain the correct watermark orientation to the precision of  $\Delta_r$ .

In [260], a rough “distance” metric was used as the basis for performance measure,

$$\text{ADE} = \frac{1}{N} \sum_{n=1}^N \sqrt{(\chi_n - \chi_{\text{actual}})^2 + (\theta_n - \theta_{\text{actual}})^2} \quad (4.5)$$

where  $\chi_n, \theta_n$  represent a scale and rotation estimate by the synchronizer and  $\chi_{\text{actual}}, \theta_{\text{actual}}$  represent the actual scale and rotation. (The ADE is defined in [260] as the average detection error.) This measure was not used in these experiments for several reasons. First, decreased ADE does not necessarily reflect better synchronization. For example, whether the synchronizer is incorrect by  $3^\circ$  or  $30^\circ$  is irrelevant if the watermark detector cannot detect the watermark in either case. Second, the belief of the author is that synchronization is generally a search problem (and with templates, an informed search problem), and a better metric for synchronization reflects the number of search attempts needed to achieve synchronization. Even if the initial searches produce poor rotation and scale estimates (resulting in a high ADE), a synchronization technique which converges on the true scale and rotation with less search is more preferable. There is no intention to state that ADE is a useless measure because small ADE is likely to result in successful synchronization (with the precise meaning of “small” dependent on the watermarking technique). However, the scale and rotation ranks quantify the success of the search more directly, and are used herein as the performance measure.

### 4.3.2 Discussion

Several experiments were performed with various attacks. The first experiment examined synchronization performance under (uniform) scaling attack, without ro-

tation. The scaling factor was varied from 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%, 110%, 120%, 130%, 140%, 150%, 160%, 180%, 200%, to 210% scaling. The scale and rotation ranks are shown in Figure 4.11. Next, synchronization performance was examined under rotation attack, without scaling. The watermarked image was rotated by angles  $0^\circ$ ,  $1^\circ$ ,  $3^\circ$ ,  $5^\circ$ ,  $15^\circ$ ,  $25^\circ$ ,  $30^\circ$ ,  $40^\circ$ ,  $45^\circ$ ,  $50^\circ$ ,  $60^\circ$ ,  $63^\circ$ ,  $70^\circ$ ,  $75^\circ$ ,  $80^\circ$ ,  $85^\circ$ ,  $87^\circ$ , and  $89^\circ$ . Results are shown in Figure 4.12. Lastly, both scaling and rotation attacks were performed. In Figure 4.13, the attack scaling factor  $f$  is varied from 0.3 to 2.1 and the rotation is fixed at  $\theta = 3^\circ$ . In Figure 4.14, the scaling is fixed to  $f = 1.2$  and the rotation is varied.

Over all of the experiments, synchronization is generally successful (with low scale and rotation ranks) for the  $M = 1$ ,  $B = 16, 32$  watermarks, the  $M = 2$ ,  $B = 8, 16$  watermarks, and the  $M = 3$ ,  $B = 8$  watermark. (Specific exceptions will be described below.) Synchronization generally fails for the  $M = 1$ ,  $B = 8$  watermark, as well as  $M = 2$ ,  $B = 32$  and  $M = 3$ ,  $B = 16, 32$  watermarks.

The synchronization failures for the  $M = 2$ ,  $B = 32$  and  $M = 3$ ,  $B = 16, 32$  watermarks are caused by an insufficient number of synchronization blocks, or equivalently, insufficient spatial redundancy. Two effects occur when there is an insufficient number of synchronization blocks: The first effect is a decrease in the the magnitude of the peaks of the autocorrelation grid, which are induced by the synchronization blocks. Decreased magnitude of the grid peaks (1) may cause some peaks to be missed entirely by the peak finder; (2) may cause more “noise” peaks to occur in the autocorrelation when the magnitude of the grid peaks are similar to or less than the magnitude of peaks arising from the original image; (3) may cause some grid peaks to be lost when the peak list is truncated to the  $P$  peaks of greatest magnitude; (4) increases the likelihood of “perturbed” peaks caused by host-signal interference or interpolation. The second effect of a reduced number of synchronization blocks is that there may not be enough peaks of the autocorrelation grid to estimate the rotation and scale in the presence of “noise” peaks, even if all of the grid peaks are precisely identified by the peak finder. The number of synchronization blocks is

identical to the total number of macroblocks in the image, which decreases as the macroblock size ( $MB \times MB$  pixels) increases.

The scale estimation performance degrades when the neighbor distance of the grid is small. This occurs for small macroblocks ( $M = 1, B = 8$ ), or when the attack scales the watermarked image to a small scaling factor (less than 50%.) There are several reasons for the decreased performance. First, when the distances between peaks become smaller, perturbations of the peak positions introduce larger relative error in the peak distances. Second, histogram smoothing increases the difficulty of precisely estimating small neighbor distances. Third, constructing the distance histogram effectively quantizes the peak distances in accordance to  $\Delta_s$ , which may increase the difficulty in estimating the neighbor distance when  $\Delta_s$  is not small compared to  $\chi$ . When the scaling factor is small, image interpolation also has a greater effect (as a removal attack) on the embedded watermark. Thus, the scale ranks rise significantly when the scale factor is less than 60% for all experiments which the scale factor is decreased.

Synchronization performance under both scaling and rotation attack is generally worse than performance under scaling or rotation alone. When the rotation is varied, difficulties in estimating the scale for rotations near  $\theta = 45^\circ$  were observed (see the sharp rise in the scale ranks of Figure 4.14 for  $M = 2$  watermarks and  $M = 3, B = 8$ .) These difficulties arise because the grid pattern is not present in the autocorrelation. For these rotation angles, a significant area of the attacked image consists of the border regions created by padding (see Figure 4.15) when the image dimensions are expanded to accommodate the rotated image. When the autocorrelation is obtained, these border regions reduce the magnitude of the peaks that form the grid while at the same time the edges between the border region and the image cause very strong peaks to occur along the edge. These border regions also cause the rise in the scale rank for  $M = 1, B = 32$  *Fruit* image, although in this case the border regions do not completely destroy the grid of peaks. One potential way to address the border region issue is to crop the watermarked image to only the

center-most area prior to obtaining the autocorrelation. Cropping should remove the large areas near the image borders and allow the autocorrelation grid to be obtained, although cropping also reduces the magnitude of the grid peaks because less of the image is used to produce the autocorrelation. Perhaps ironically, when the autocorrelation peaks become dominated by the border regions, their arrangement makes rotation estimation very easy. Hence, rotation estimation is not affected like the scale estimation.

The fact that two bicubic interpolation operations are applied to the watermarked image when both scaling and rotation attacks are performed is also likely to have an effect on the synchronization performance. Each bicubic interpolation may damage the watermark, thereby reducing the magnitude of the peaks on the grid. In particular, the effects of padding were observed when the attack consists of both scaling and rotation, but were not observed when rotation was performed without scaling.

Comparing amongst the images, the *Crowd* image is most difficult to synchronize and the *Girl* image is the easiest. The structure of the *Crowd* image has many peaks in the autocorrelation, which introduces host-signal interference to the synchronizer. The effect of this interference is an increased number of “noise” peaks, which is most obvious by the increased scale ranks of *Crowd* when the scale factor is reduced. Often, the scale rank of *Crowd* is much worse than the other images when the scale factor is less than 100%.

In summary, the proposed synchronization technique is successful under uniform scaling and rotation attack for the  $M = 1, B = 16, 32$  watermarks, the  $M = 2, B = 8, 16$  watermarks, and the  $M = 3, B = 8$  watermark, although there is an issue in the scale estimation when an attacked image is rotated near  $45^\circ$  caused by padding. Also, very small neighbor distances are difficult to estimate, which limits the size of macroblocks and makes estimation of very small scale factors difficult. Despite these limitations, the spatial synchronization mechanism (the rotation and scale estimation) is generally successful in synchronizing under uniform scaling and

rotation attack when there is sufficient spatial redundancy to induce a regular grid pattern of peaks in the autocorrelation.





(a) Original *Girl*



(b) Watermarked *Girl*



(c) Original *Crowd*



(d) Watermarked *Crowd*

Fig. 4.9. Original and watermarked *Girl* and *Crowd* ( $\gamma = 5.0$ ,  $M = 2$ ,  $B = 16$ )

(a) Original *Fruit*(b) Watermarked *Fruit*(c) Original *Peppers*(d) Watermarked *Peppers*Fig. 4.10. Original and watermarked *Fruit* and *Peppers* ( $\gamma = 5.0$ ,  $M = 2$ ,  $B = 16$ )

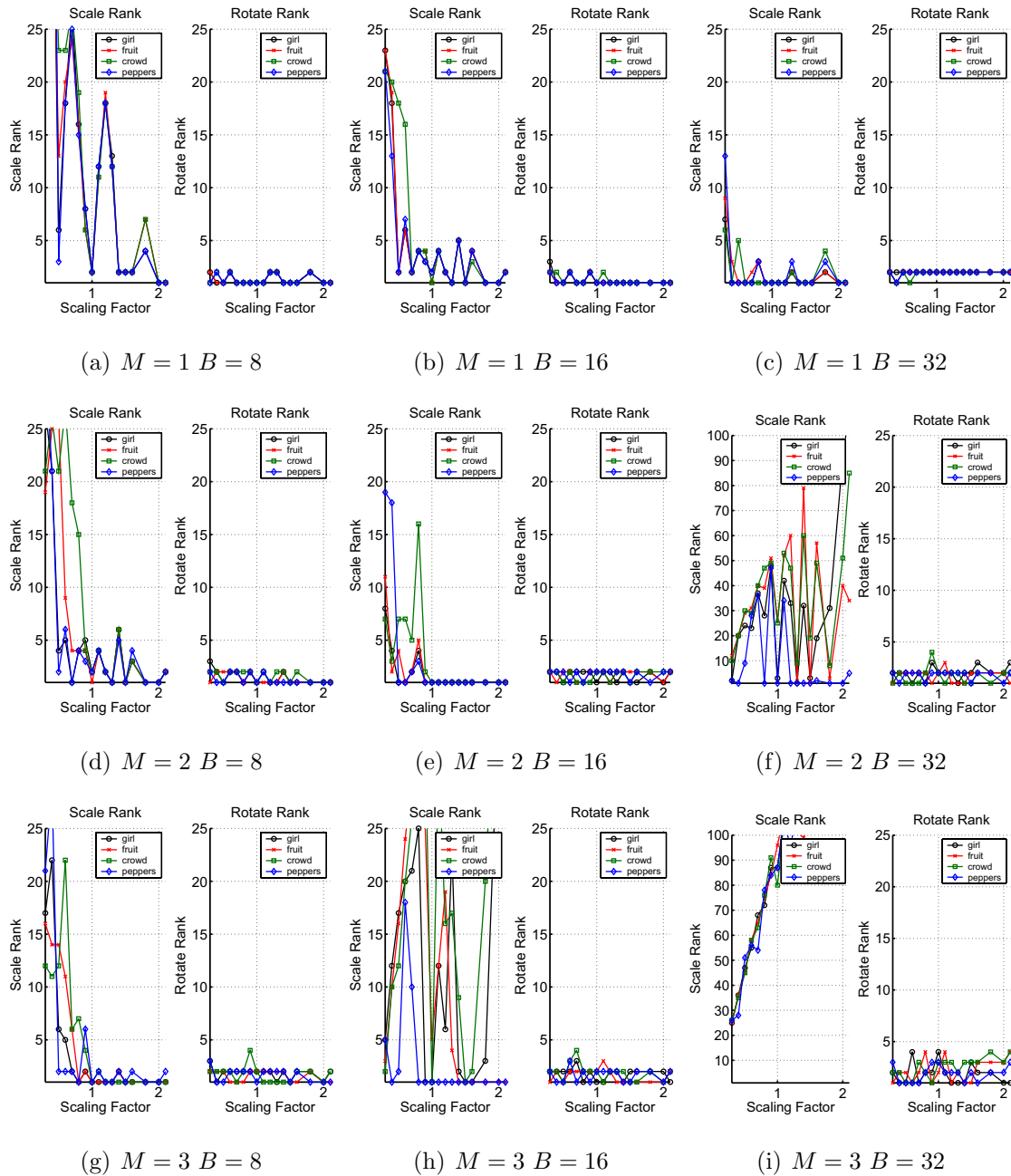


Fig. 4.11. Scale and rotate ranks for scaling attack (30% to 210%) without rotation, with watermark construction parameters  $M$  and  $B$  varied.

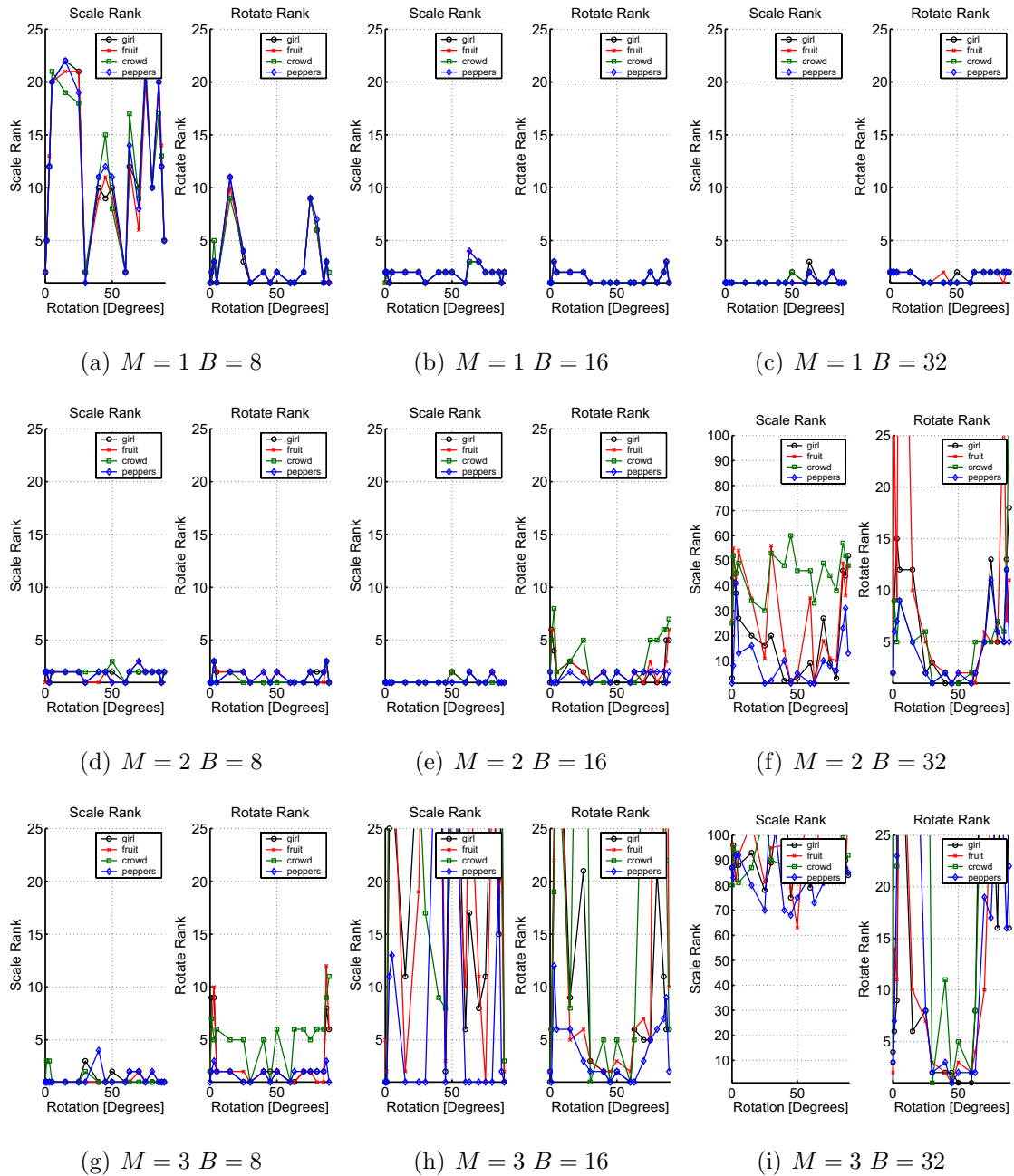


Fig. 4.12. Scale and rotate ranks for rotation attack ( $0^\circ$  to  $89^\circ$ ) without scaling, with watermark construction parameters  $M$  and  $B$  varied.

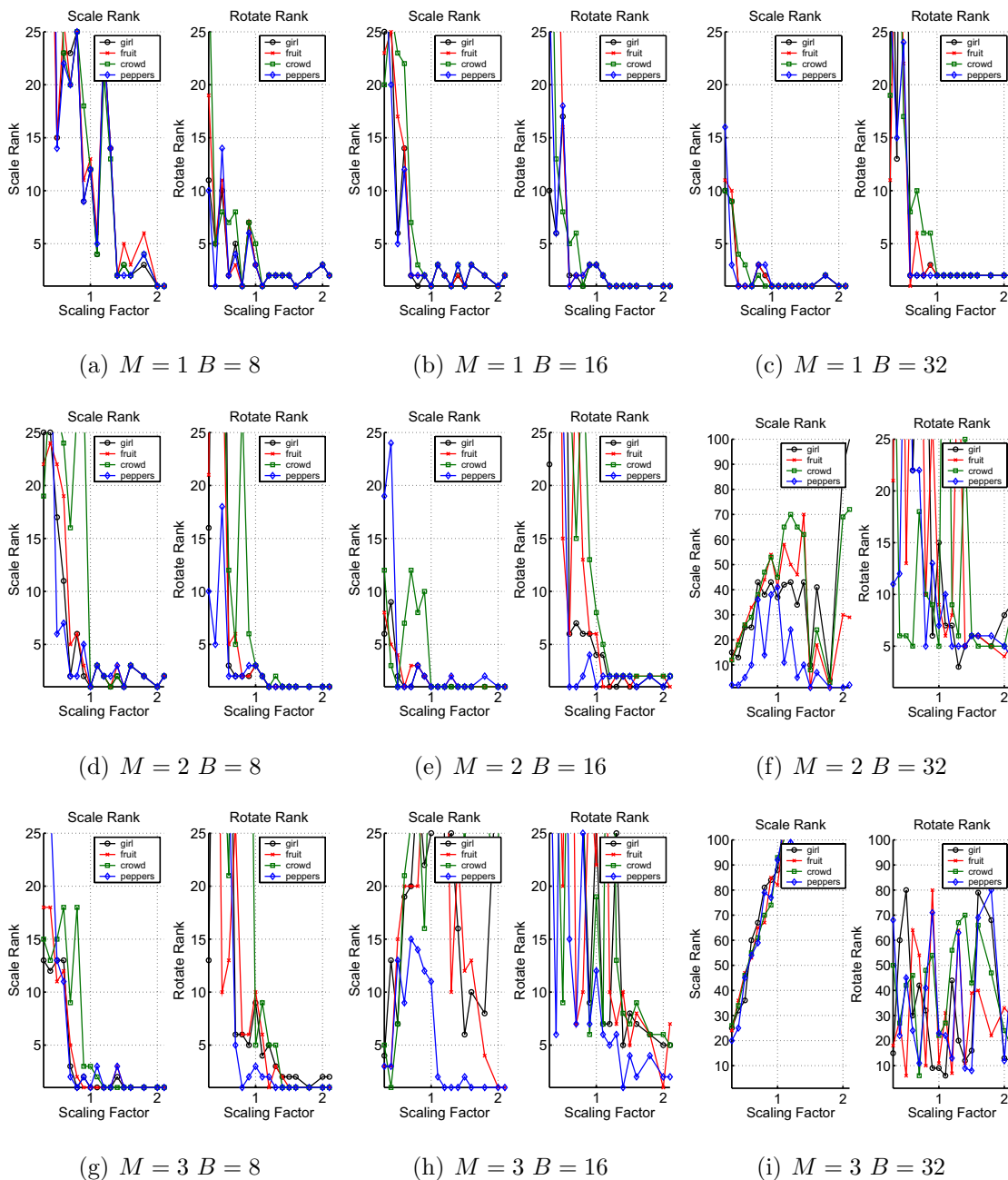


Fig. 4.13. Scale and rotate ranks for rotation and scaling attack. Scaling is varied from 30% to 210% while rotation is fixed at  $3^\circ$ . Watermark construction parameters  $M$  and  $B$  varied.

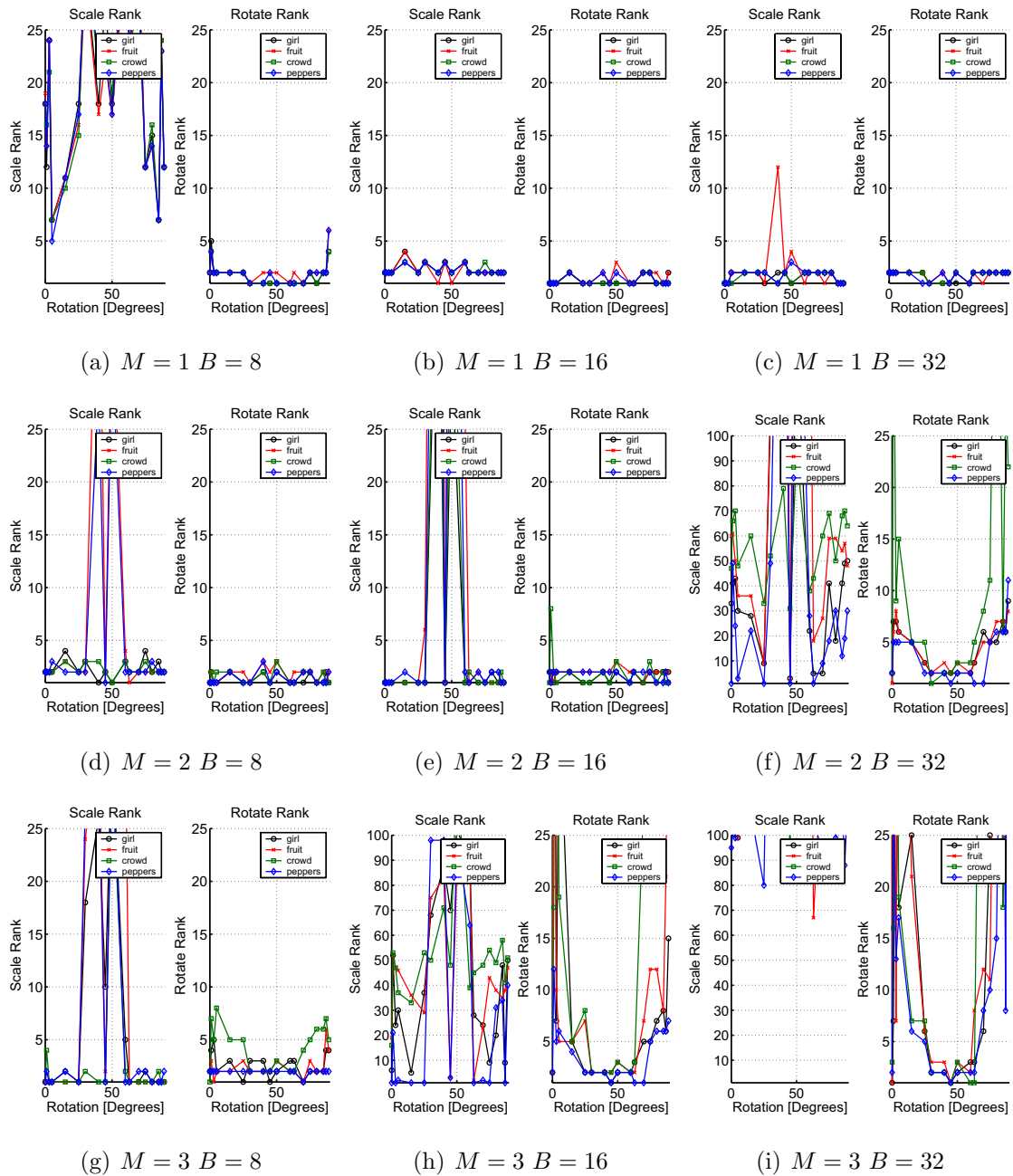
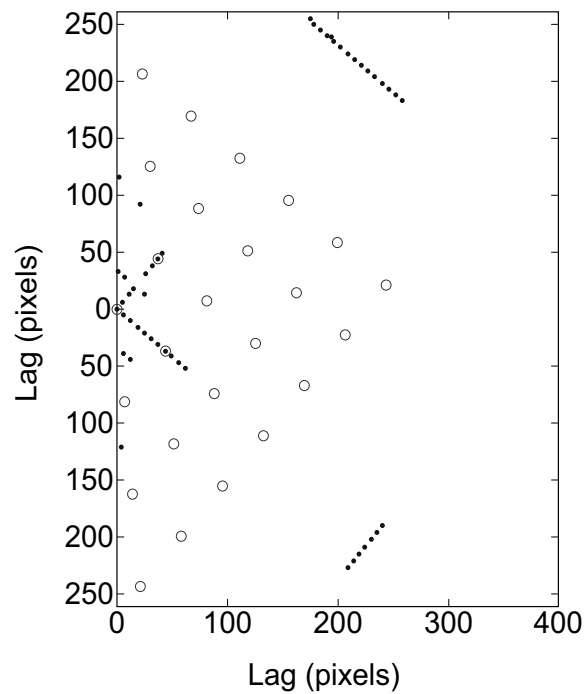


Fig. 4.14. Scale and rotate ranks for rotation and scaling attack. Scaling is fixed at 120% while rotation is varied from  $0^\circ$  to  $89^\circ$ . Watermark construction parameters  $M$  and  $B$  varied.



(a) Attacked image (*Girl* Scaling 120% Rotation 50°)



(b) Autocorrelation of attacked image. Dots (·) indicate observed peak positions, circles (○) indicate expected peak positions for the grid-like pattern, which is absent.

Fig. 4.15. Effect of image padding. When an image is rotated near 45°, a significant area of the attacked image is occupied by padding (the black triangular regions.) These regions, and the strong edges between the padded regions and the image, disrupt the pattern of peaks in the autocorrelation.

#### 4.4 Conclusions and Future Work

In this chapter, a technique was described for spatial synchronization for still images and video frames. The watermark was constructed in a manner inspired by the temporal synchronization technique of Chapter 3. For spatial synchronization, redundant blocks are used to construct an array of peaks in the autocorrelation. The synchronization method examines the arrangement of the peaks to estimate the position and orientation of the watermark. Like the work in temporal synchronization, insufficient redundancy in the watermark led to poor performance of the synchronizer. This technique is applicable for synchronization under uniform scaling and rotation attack.

To improve the performance of the technique, subsequent efforts should consider reducing host-signal interference [269] and improving the precision of the estimation. It was noted that peaks in the the autocorrelation may be perturbed from their ideal locations. Some of the perturbations of the peak positions occur because peak locations are discretized to pixel locations by the (simple) peak finder. This “quantization” effect on the peak locations limits the precision by which the rotation and scale can be estimated [297]. In addition, the effects of image padding are profound when the rotation is near  $\theta = 45^\circ$ . Another method for improved effectiveness, particularly for images such as *Crowd*, is to adapt the orientation of the watermark to the characteristics of the original image. For example, the embedder can detect existing peaks in the autocorrelation of the original (unwatermarked) image and then “align” the watermark (for example, rotate and scale the watermark) in accordance to the peaks.

Although this synchronization technique exploits geometric properties of the grid of peaks that would be challenging to generalize to other attacks, we desire to find a more general framework for spatial synchronization that will allow the modeling of spatial synchronization attacks other than uniform scaling and rotation. A more formal method to quantify (spatial) redundancy is desired. The construction of the



watermark described in Section 4.1 was inspired by the temporal synchronization work, however spatial redundancy can be induced by other watermark construction methods. For example, a watermark designed as a tessellation has a redundant or regular pattern, but not necessarily in the form of macroblocks. Embedding shifted versions of the same signal, as in [267], is also a method for introducing redundancy. This work shows that spatial redundancy is helpful for spatial synchronization, but we seek to obtain a more general framework that takes this work, as well as [260, 266, 267], as special cases.

## 5. SEMI-FRAGILE WATERMARKING

This chapter describes earlier work in semi-fragile watermarking, which is not related to watermark synchronization. Fragile and semi-fragile watermarks have considerably different properties and applications than the robust watermarks discussed in the earlier chapters. A semi-fragile watermarking technique is described which allows detection of alterations to a watermarked image but allows the watermarked image to undergo lossy JPEG compression.

### 5.1 Robust, Fragile, and Semi-Fragile Watermarking

The discussion of watermarking thus far has focused on robust watermarks, which are watermarks that are designed to be detectable even if the watermarked signal has been attacked. Robust watermarks are proposed for applications such as content tracking, copyright watermarking, copy protection, and broadcast monitoring, where embedded watermark should be difficult to remove. A robust watermark should be a permanent and inseparable part of the watermarked signal.

Unlike robust watermarks, fragile watermarks [29, 41, 42, 44, 190, 195, 220, 298, 299] are not designed to be robust against attacks. On the contrary, a fragile watermark is designed to be destroyed by even the slightest alteration or modification to the watermarked signal. This property of fragile watermarks is useful for authentication applications, where the objective is to provide confidence that a signal (1) originated from a known source, and (2) has not been tampered or altered. For example, a digital camera can embed a camera-dependent signature (the watermark) into an image as the user takes a snapshot [40], which allows the user to demonstrate that the image was taken using the camera. Authenticating digital images, video, and other content is an important and challenging problem because digital data can be

easily manipulated. This limits the credibility of digital data as evidence, particularly in surveillance [39], law, journalism, intelligence, and other applications where the trustworthiness of the data must be demonstrated.

The authenticity of a watermarked signal can be obtained by using the watermark detector. If the detector successfully detects the watermark, both the source of the signal and the integrity of the signal are authenticated. The source of the signal is authenticated because the watermark was embedded with secret  $K_E$ , which should be available only to the source. The integrity of the signal is authenticated because tampering will damage or destroy the embedded fragile watermark. If the watermark is not detected, then either the signal did not originate from the source (that is, the signal may be a forgery) or the entire signal had been tampered with. The embedded watermark could be damaged, but not completely destroyed, which could indicate localized tampering. This leads to the next benefit of fragile watermarking: localization.

In addition to tamper detection, fragile watermarking allows tampering to be localized. Localization is identifying regions of the signal that are likely tampered and distinguishing these regions from regions that have not been tampered. The ability to localize tampering is an advantage of fragile watermarking compared with other authentication methods, for example digital hashes or signatures [16,292,294]. While a digital hash or signature can easily identify altered signals, localization requires significantly more effort. For example, localization using digital signatures is possible by signing features obtained from the signal [300] (as opposed to signing the signal directly) or by partitioning a large signal into smaller regions and signing each region individually. Also, a digital signature is separate from the signal to be authenticated. Keeping the signature and signal together may be cumbersome and the signature may be prone to becoming lost. Whereas an embedded watermark cannot be accidentally separated from the signal to be authenticated. Some fragile watermarks use or embed digital hashes to obtain the desired fragileness properties [41].

One of the drawbacks of fragile watermarking is that the fragileness of the watermark may be *too* sensitive for some applications. For example, if a lossy compression technique, such as JPEG [54], was applied on a watermarked image, then the fragile watermark detector will report that the entire image has been tampered with even though the compressed image appears nearly the same as the watermarked image. This is not desired in some applications. It would be nice if the watermark detector could identify “information-altering” transformations, which change the watermarked signal in a significant way, but not be sensitive to “information-preserving” transformations, which change the signal in a literal sense but not in a way that is significant to the application. Watermarks with fragileness to some transformations but robustness to other transformations are known as semi-fragile watermarks [43, 220, 301–308].

The objective of attacks against fragile and semi-fragile watermarks are different than attacks against robust watermarks. Removal attacks and detection-disabling attacks prevent a robust watermark from protecting the content by removing or obfuscating the watermark. However, these attacks do not affect fragile and semi-fragile watermarks. Quite the contrary— if a removal or detection-disabling attack was applied against a fragile or semi-fragile watermark, the watermark detector would report the watermarked signal as unauthentic. Because these attacks involve tampering with the watermarked signal, the assessment by the detector is correct. While removal and detection-disabling attacks are generally not a concern for (semi-)fragile watermarks, counterfeiting and forgery attacks [309, 310], such as the copy attack, are much more significant. These attacks may allow the attacker to create a falsified signal that appears authentic to the watermark detector.

There is some irony in using watermarks for authentication. The watermark detector cannot determine the authenticity of the original signal because the original signal is not watermarked. The watermark detector can determine the authenticity of the watermarked signal, but watermark insertion introduces distortion into the original signal. Protecting the authenticity of a signal by using (semi-)fragile wa-

termarking requires watermark insertion, which could be viewed as tampering. To address this concern, some techniques allow the original signal to be recovered by the watermark detector if it determines that its input is authentic. That is, the watermark detector can obtain  $X$  if its input  $Z$  is the watermarked signal  $Y$ . However, if the input is an altered signal  $Z = \hat{Y}$ , then  $X$  is generally not recoverable. These techniques are known as reversible or invertible watermarks [311–315].<sup>1</sup>

### 5.1.1 Semi-fragile watermarking

A semi-fragile watermark combines the properties of fragile and robust watermarks. Like a robust watermark, a semi-fragile watermark is capable of tolerating some degree of change to the watermarked signal, such as the addition of quantization noise from lossy compression. These changes are known as “information preserving” transformations because they do not substantially affect the use of the watermarked signal in the application. And like a fragile watermark, the semi-fragile watermark is capable of localizing regions of the image that have been substantially tampered by “information altering” transformations. For authentication applications, blind detection is almost always required. In addition, the distortion introduced by watermark insertion also should not degrade the value of the image in the application.

The application defines which transformations or processes are “information-preserving” and which transformations are “information-altering.” Feature replacement is the (deliberate) alteration of specific areas of a signal and is usually considered information-altering. Examples of feature replacement in digital images include: altering or juxtaposing faces or bodies of persons shown in the image; editing text shown on the image, such as a sign or license plate; removing an object from the image; inserting an object into the image. Applications may consider lossy compression to be information-preserving. Other examples of transformations that may either be information-altering or information-preserving include global amplitude adjustment

---

<sup>1</sup>In this context, “invertible” does not have the same meaning as when discussing ambiguity attacks.

(such as the volume for digital audio, or brightness for images), contrast adjustment, gamma correction [136], digital-to-analog(-to-digital) conversion, colorspace transformation [136] or color-to-grayscale conversion (for images), stereo-to-mono conversion (for audio), transcoding [73], and cropping.

In the development of a semi-fragile watermarking technique, one should consider the challenges that prevent naïve use of a good fragile or robust watermarking technique as a semi-fragile technique. Many fragile watermarking techniques insert the watermark into the least-significant bit (LSB) plane of a signal and are unable to tolerate a single bit error in the LSB. However, noise introduced into the watermarked signal by information-preserving transformations is likely to cause many LSBs to change. Secondly, cryptographic hash functions used by some fragile watermarking techniques are challenging to use in a semi-fragile watermark. To use a hash function for semi-fragile watermarking, the hash would need to be obtained over some characteristic of the signal that is invariant to information-preserving transformations. The output of the hash function (or the digest) may also need to be embedded in a way that is resilient to noise introduced by information-preserving transformations.

The challenge of transforming a robust watermarking technique into a semi-fragile technique is localization. Robust watermarking techniques generally obtain their robustness properties by generating the watermark as a long signal. Long signals are more resilient against noise introduced by attacks and is also helpful to the watermark detector in overcoming host-signal interference. However, while using longer sequences improve watermark detection performance, long sequences also limit localization.

Many papers proposing semi-fragile watermarking techniques suggest JPEG compression as an information-preserving transformation. JPEG is a lossy compression technique for still images [54] that is often used for digital photography and Internet imagery. To control the compressor, applications often specify a “quality factor” or quality parameter  $Q$  which affects the degree of quantization internally used by the compressor to encode an image. The typical range of  $Q$  is from 1 to 100. Low

values of  $Q$  produce encoded images that are smaller in size but have significant amounts of distortion introduced as quantization error. High values of  $Q$  introduce relatively little distortion, but the size of the encoded image is larger. As a general rule, low values of  $Q$  produce “low quality” images and high values of  $Q$  produce “high quality” images.

Embedding multiple watermarks [316] has also been proposed as a method of authenticating an image with a degree of robustness. For example, a robust watermark can be embedded into an image to establish ownership followed by the embedding of a fragile watermark for authentication. The primary disadvantage is that if lossy compression or other information-preserving transformations are performed, then most or all of the authentication information is lost whereas a semi-fragile watermark is capable of providing some confidence of authenticity even after information-preserving transformations.

## 5.2 A Semi-Fragile Watermark

The semi-fragile watermarking technique described in this chapter (see also [43]) is based on extending a spread-spectrum watermarking technique with a modified detector that correlates pixel value differences in the spatial domain. The detection process is repeated for blocks of an image, allowing alterations to be identified and localized.

### 5.2.1 Watermark generation and insertion

The watermark is constructed in the DCT domain as smooth patterns that will resist being damaged by JPEG compression. The watermark is constructed as a signal with the same size (dimensions) as the original image, partitioned into non-overlapping blocks of fixed size. For each block, a PRNG is used to produce a zero-mean unit variance Gaussian distributed signal. A different pseudo-random sequence is generated for each watermark block, and the embedding key  $K_E$  is used

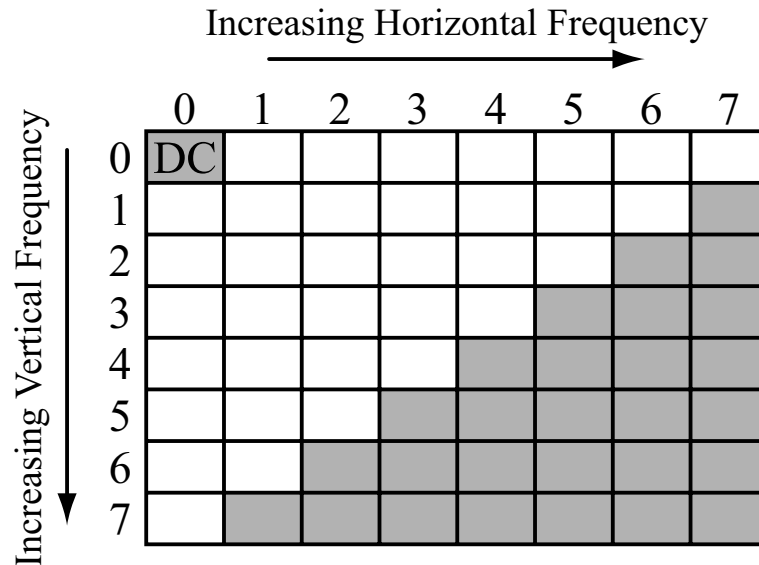


Fig. 5.1. Watermark generation in DCT domain for  $8 \times 8$  blocks. For each watermark block, white coefficients are generated using a PRNG. Shaded coefficients are not watermarked and have a value of zero.

to seed the PRNG prior to constructing the first watermark block. The DCT uses cosines as the basis functions for the transformation and each coefficient corresponds to basis functions of different spatial frequencies. Only the middle frequencies of the DCT coefficients are watermarked; see Figure 5.1. The DC coefficient is not watermarked because watermarking this coefficient is likely cause visible block artifacts in the watermarked image. Watermarking the DC coefficient also would not contribute significantly to the performance of the detector. The high frequency AC coefficients are not watermarked because these coefficients are likely to be affected by JPEG compression. Once the watermark has been constructed in the DCT domain, applying the inverse DCT produces a spatial domain watermark  $W$ .  $W$  is then inserted into the original image  $X$  via additive embedding, or equation (2.1), to produce the watermarked signal  $Y$ . For color images, the watermark is embedded in the luminance.



$$B(x, y) = \begin{bmatrix} -1 & 1 & 4 & -7 \\ 3 & 3 & 5 & -1 \\ 5 & 1 & 4 & 3 \\ 1 & -4 & -5 & -3 \end{bmatrix} \rightarrow \left\{ \begin{array}{l} \Delta_{COL}(B) = \begin{bmatrix} -2 & -3 & 11 & 0 \\ 0 & -2 & 6 & 0 \\ 4 & -3 & 1 & 0 \\ 5 & 1 & 2 & 0 \end{bmatrix} \\ \Delta_{ROW}(B) = \begin{bmatrix} -4 & -2 & -1 & -6 \\ -2 & 2 & 1 & -4 \\ 4 & 5 & 9 & 6 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \right.$$

Fig. 5.2. Example of block operators on a block  $B$  of size  $4 \times 4$  samples

### 5.2.2 Watermark Detection

The detector is based on the differences of adjacent pixel values in the spatial domain. Most natural images consist of large areas of relatively smooth features with an occasional edge, so the power in a difference signal should be substantially less than the power of the signal itself. Therefore, unless an edge is present, the difference between adjacent pixel values of the watermarked signal will be the embedded watermark signal with relatively low amount of host-signal interference. Define an operator  $\Delta_{COL}(\cdot)$  as the difference-of-columns, where  $B(x, y)$  is an arbitrary block and  $BS$  is the blocksize:

$$\Delta_{COL}(B(x, y)) = \begin{cases} B(x, y) - B(x + 1, y) & \text{if } x \in \{1, \dots, BS - 1\} \\ 0 & \text{if } x = BS \end{cases} \quad (5.1)$$

The difference-of-rows operator  $\Delta_{ROW}(\cdot)$  can be similarly defined:

$$\Delta_{ROW}(B(x, y)) = \begin{cases} B(x, y) - B(x, y + 1) & \text{if } x \in \{1, \dots, BS - 1\} \\ 0 & \text{if } x = BS \end{cases} \quad (5.2)$$

Figure 5.2 shows the effect of applying these operators for an example block.

Let  $Z_b$  be a block of the test image provided to the detector and  $W_b$  be the corresponding block of the watermark (in the spatial domain), where  $b$  is the block

index. The detector can regenerate  $W_b$  given the detection key and embedding strength. The watermark detector will correlate both the row and column differences. Let  $Z_b^*$  be defined as the concatenation of the column-difference and row-difference of the test block and let  $W_b^*$  be the corresponding differences of the watermark block as shown in (5.3) and (5.4) below:

$$Z_b^* = \begin{bmatrix} \Delta_{COL}(Z_b(x, y)) & \vdots & \Delta_{ROW}(Z_b(x, y)) \end{bmatrix} \quad (5.3)$$

$$W_b^* = \begin{bmatrix} \Delta_{COL}(W_b(x, y)) & \vdots & \Delta_{ROW}(W_b(x, y)) \end{bmatrix} \quad (5.4)$$

Correlating  $Z_b^*$  and  $W_b^*$  involves  $2(n^2 - n)$  points (where the subtraction of  $n$  reflects the row or column of zeroes after the difference operator), which is greater than the  $n^2$  points for a spatial correlation of the block. By correlating the spatial difference between adjacent pixels as opposed to a direct correlation, host-signal interference during watermark detection is reduced, provided that the image is sufficiently smooth in the block.

The block detection statistic is then the normalized correlation

$$\rho_b = \frac{Z_b^* \cdot W_b^*}{\sqrt{(Z_b^* \cdot Z_b^*)(W_b^* \cdot W_b^*)}}. \quad (5.5)$$

The dot product is defined on vectors and not matrices, so  $Z_b^*$  and  $W_b^*$  are first “reshaped” to a row or column vector. The permutation by which the reshaping is performed is not important but the same permutation must be used for both  $Z_b^*$  and  $W_b^*$ . Once the normalized correlation statistic  $\rho_b$  has been obtained for each block, it is compared to a threshold  $T$  to determine the classification for each block:

$$\text{Detector Output} = \begin{cases} \rho_b > T & : \text{Block is authentic} \\ \rho_b < T & : \text{Block is altered} \end{cases} \quad (5.6)$$

The framework of (5.5) and (5.6) is that of a binary hypothesis test between the hypotheses

$$\begin{aligned} H_0: & \text{Block is authentic} \\ H_1: & \text{Block is altered} \end{aligned} \quad (5.7)$$

Defining the conditional means

$$\rho_U^* = E[\rho_b \mid \text{Block } b \text{ is authentic}] \quad (5.8)$$

$$\rho_A^* = E[\rho_b \mid \text{Block } b \text{ is altered}] \quad (5.9)$$

then the performance of the detector is based on the mean separation or  $|\rho_A^* - \rho_U^*|$ . The larger the mean separation, the better the performance of the detector. A more detailed analysis, such as obtaining the probability of false positive and miss, will require making assumptions regarding the conditional distributions (and not merely the expectations) under each hypothesis as well as the a priori probabilities for a Bayesian framework. For example, if conditional Gaussian distributions are assumed for convenience, then the detector performance can be expressed as a function of  $\rho_A^*$ ,  $\rho_U^*$ , and their respective variances using well-known results from statistical hypothesis testing [100, 101, 103].

### 5.3 Evaluation of the Semi-Fragile Watermark

A synthetic image *Gradient* and real images *Girls*, *Sign*, and *Money* were altered to evaluate the semi-fragile watermarking technique. The original images are shown in Figure 5.3 and the altered images are shown in Figure 5.4. For each pair of original and altered images, a difference image is constructed. The difference image is necessary to evaluate the performance of the watermark detector, but the watermark detector itself will not have access to the difference images. Neither the original, altered, nor difference images are watermarked.

The watermark embedder is provided with each original image to produce the watermarked image (with  $\gamma = 5.0$ , randomly chosen  $K_E$ ). Then, for every pixel where the original and altered images differ, the pixel value in the watermarked image is replaced with the corresponding pixel value in the altered image, producing a tampered image. This simulates an attacker acquiring a watermarked image and altering it. Finally, the tampered image is compressed using JPEG with various quality factors.

As mentioned in Section 5.2.2, the values of  $\rho_b$  for altered and unaltered blocks determine the detectability of alterations. Let  $\rho_U$  be the set  $\{\rho_b\}$  for all unaltered blocks and  $\rho_A$  be the set  $\{\rho_b\}$  for all blocks containing at least one altered pixel. Table 5.1 below shows the observed values for  $E[\rho_U]$ ,  $E[\rho_A]$ ,  $\text{Var}[\rho_U]$ , and  $\text{Var}[\rho_A]$ . Figure 5.5 shows the same results in graphical form. The block size is  $16 \times 16$  pixels for all images.

The detector has little difficulty discerning the unaltered and altered blocks for lightly compressed *Gradient* and *Girls* images, as the difference in the mean correlations of altered and unaltered blocks is large. However, effects of the edges in *Sign* and the textures in *Money* can be seen by the low difference of means for even high-quality compression. Out of the three real images examined, the best performance is seen in *Girls* and the worst performance in *Money*.

The detector performance, measured as the percentage of correctly classified blocks, varies as the detection threshold is changed. Figure 5.6 shows the percentage of correct detections for the four images at various compression levels (see Table 5.1 for the data rates at each JPEG quality factor). For moderate compression, at least 75% correct block detection was achieved for all images using a threshold of  $T = 0.1$ . It is interesting to note that the performance does not decrease uniformly as the amount of compression is increased; some edges present in the image that cause detector errors may be “softened” by lossy compression, yielding better detection.

Figure 5.7 shows an example illustrating the performance of the detector (see the caption for the embedding and detection parameters). Most of the detection errors occur near the edges in the image or in textured areas. Many detection misses occur in blocks which contain a boundary between an altered region and unaltered regions. A block is considered “altered” even if a single pixel value within that block was changed from the original image. However, a semi-fragile detector with single-pixel sensitivity and resilience to lossy compression may not be feasible.

Table 5.1  
Block Statistics for detector (embedding strength  $\gamma = 5.0$ , detection  
blocksize= $16 \times 16$ )

JPEG Compress Q (%)	Image Data Rate (bits/pixel)	Unaltered Blocks			Altered Blocks		
		Actual Number of Unaltered Blocks Present	Sample Mean $\rho_b$ of Unaltered Blocks	Sample Variance of $\rho_b$ for Unaltered Blocks	Actual Number of Altered Blocks Present	Sample Mean $\rho_b$ of Altered Blocks	Sample Variance of $\rho_b$ for Altered Blocks
<b>GRADIENT</b>							
Uncompress	24	382	0.9779	0.00257	300	0.1106	0.00037
90	1.1173	382	0.8634	0.00227	300	0.0962	0.00032
70	0.5504	382	0.4983	0.00131	300	0.0477	0.00016
50	0.3617	382	0.3100	0.00081	300	0.0290	0.00010
30	0.2425	382	0.1199	0.00031	300	0.0131	0.00004
<b>GIRLS</b>							
Uncompress	24	4753	0.5608	0.00012	951	0.0670	0.00007
90	2.1843	4753	0.5452	0.00011	951	0.0646	0.00007
70	1.1072	4753	0.3337	0.00007	951	0.0385	0.00004
50	0.7627	4753	0.2313	0.00005	951	0.0239	0.00003
30	0.5072	4753	0.1322	0.00003	951	0.0114	0.00001
<b>SIGN</b>							
Uncompress	24	1459	0.2377	0.00016	77	0.1598	0.00210
90	2.9761	1459	0.2210	0.00015	77	0.0692	0.00091
70	1.3825	1459	0.1990	0.00014	77	0.0561	0.00074
50	0.8617	1459	0.1857	0.00013	77	0.0399	0.00053
30	0.5726	1459	0.0954	0.00007	77	0.0124	0.00016
<b>MONEY</b>							
Uncompress	24	427	0.2407	0.00057	143	0.0189	0.00013
90	3.7061	427	0.2330	0.00055	143	0.0174	0.00012
70	2.0879	427	0.1699	0.00040	143	0.0119	0.00008
50	1.5433	427	0.1338	0.00031	143	0.0139	0.00010
30	1.1203	427	0.1002	0.00024	143	0.0104	0.00007

(a) *Gradient*(b) *Girls*(c) *Sign*(d) *Money*

Fig. 5.3. Original Images



(a) *Gradient*



(b) *Girls*



(c) *Sign*



(d) *Money*

Fig. 5.4. Altered Images

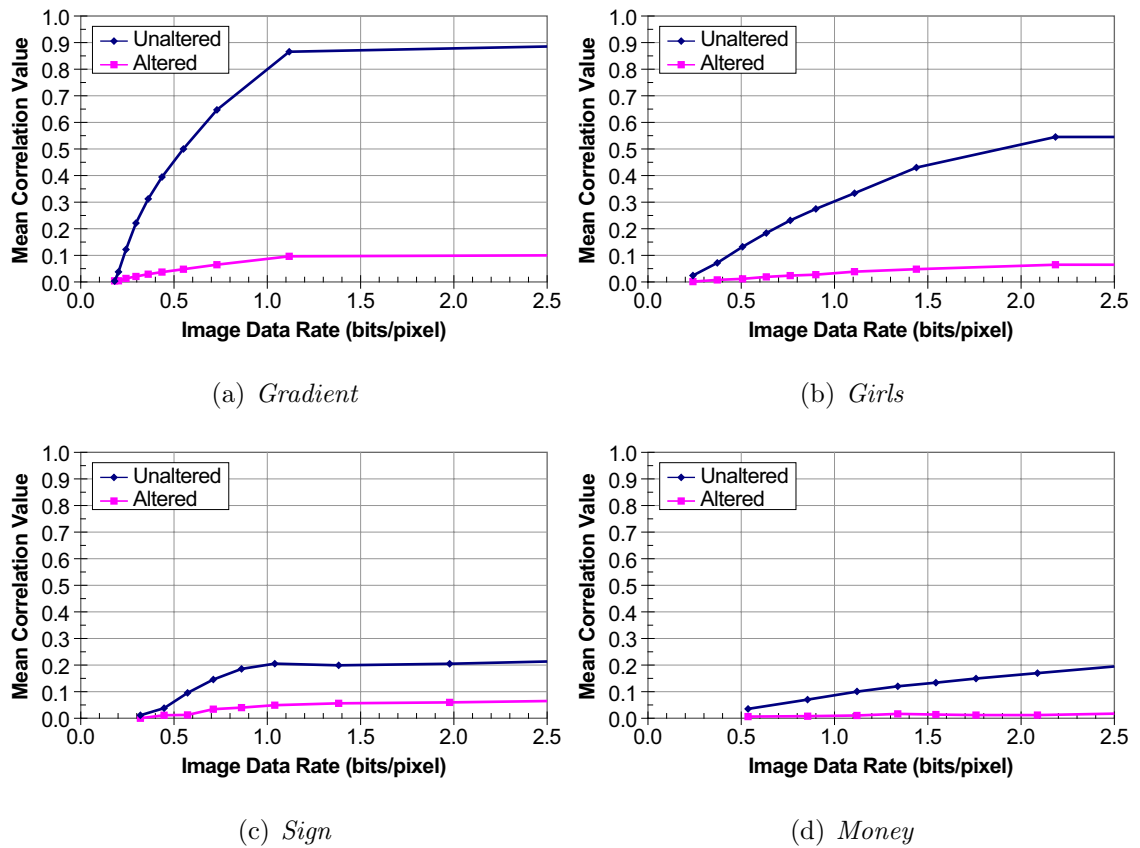


Fig. 5.5. Mean correlation of unaltered and altered blocks when the tampered image is provided to the watermark detector (embedding strength  $\gamma = 5.0$ , blocksize= $16 \times 16$ ) after varying degrees of JPEG compression. The accuracy of the detector improves when there is a large separation between the mean  $\rho_b$  of altered and unaltered blocks.



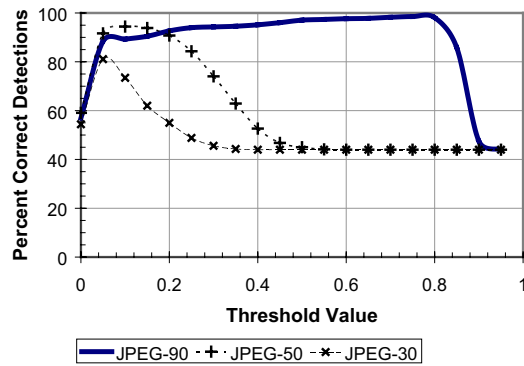
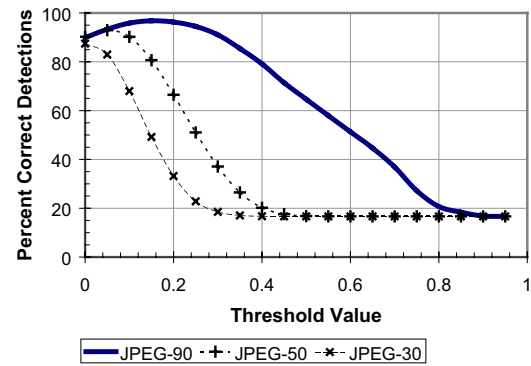
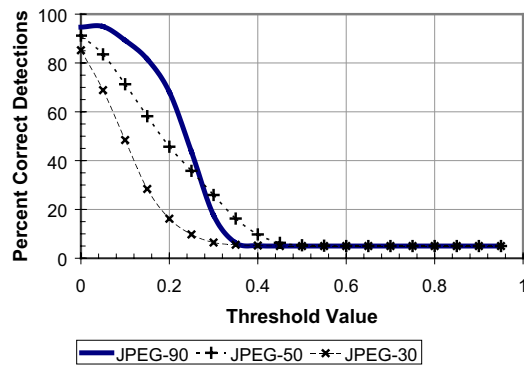
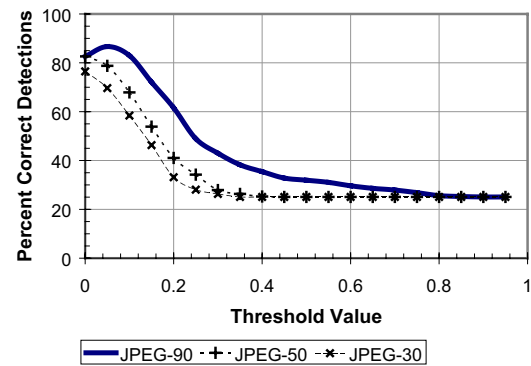
(a) *Gradient*(b) *Girls*(c) *Sign*(d) *Money*

Fig. 5.6. Detector accuracy when the threshold is varied from 0.0 to 1.0, for a tampered image compressed at various JPEG quality factors. When the threshold is near 0.0, almost all of the incorrect detections are misses. When the threshold is near 1.0, almost all of the incorrect detections are false positives.

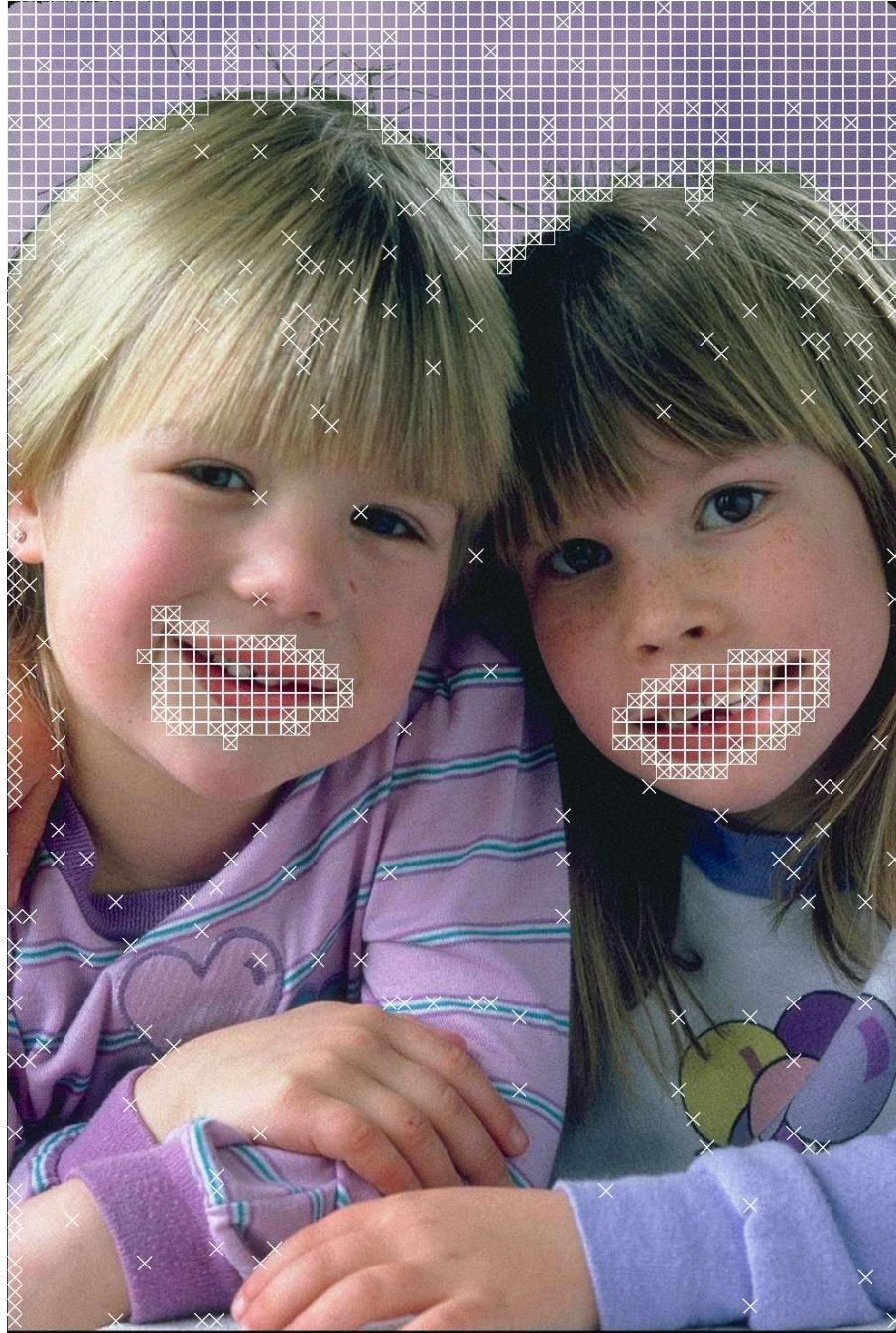


Fig. 5.7. Example of Detection.  $\gamma = 5.0$ ,  $T = 0.1$ , blocksize= $16 \times 16$ , JPEG  $Q = 60$  data rate=0.90 bits/pixel, 93% correct detection (5332 blocks correct out of 5704 blocks), 4% false positive (211 false positives out of 4753 unaltered blocks), 17% misses (161 misses out of 951 altered blocks). A box indicates an altered block correctly identified, X indicates false positive, and X within a box indicates a miss.

## 5.4 Conclusions

A semi-fragile watermark was described which could identify altered regions within a watermarked image with 75% accuracy under moderate compression and with near 90% accuracy under light compression for blocks of  $16 \times 16$  pixels. The detector was based on correlation of spatial-domain pixel differences, which takes advantage of the fact that most natural images consist of large regions that are relatively smooth. The watermark pattern itself is smooth, consisting of low and middle frequency DCT components to resist damage from JPEG compression. Edges and textures increase the likelihood of false positives and misses. While the presence of edges and texture can be a problem for the detector, regions of the image that are highly textured can also be more strongly watermarked due to the masking effects of the human visual system. Thus, perceptual shaping can be used to improve the performance of the detector.

The block size of  $16 \times 16$  is also fairly small and using larger block sizes may reduce the effect of host-signal interference, with the tradeoff of reduced precision in localization. However, tamper detection on a coarser scale (or using larger block sizes) may be sufficient for applications, particularly for reduced detection error. The performance using larger block sizes and different embedding strengths should be investigated in additional development of the technique.

The performance of this technique has been independently evaluated in [317] and was shown to have favorable robustness against non-malicious signal processing as well as good false alarm performance in comparison with other semi-fragile watermarking techniques. In this evaluation, the detector classifies each block of  $64 \times 64$  pixels as altered or unaltered, however this is not a direct extension of the technique to larger block sizes. Instead, each  $64 \times 64$  block is constructed as 16 blocks of  $16 \times 16$  pixels in size, and the entire (large) block is considered altered when the detector classifies five or more of the small blocks as altered. This scheme was used to compare different semi-fragile techniques because some techniques cannot function

at particular block sizes. It was noted in [317] that Wiener filtering (to enhance the watermark signal) may improve detector performance in this technique.

This watermarking technique was proposed prior to the copy attack [88], and the technique may be vulnerable to the copy attack. In particular, the watermark signal is generated independent of the original image. However, introducing dependence of the original image in watermark generation can provide resilience against the copy attack. For security, each block of the watermark should depend on the embedding key, the original image, as well as the position of the block within the image [309]. Fridrich also suggests generating the watermark with dependence on ancillary data, such as the image dimensions, to make attacks involving multiple watermarked signals (including the copy attack and [310]) more difficult.

## 6. CONCLUSIONS

Spatial and temporal synchronization were examined in this dissertation. For temporal synchronization, a new framework was developed that encompasses the behavior of a large number of blind symmetric video watermarking techniques. The framework expresses the temporal structure of the watermark using a state machine key generator but is agnostic to the structure of the watermark that is embedded into each video frame. The framework demonstrates a relationship between temporal redundancy and synchronization, and explains why temporal synchronization is easy for some watermarks and much more difficult for others. The framework also leads to the design of watermarks that have arbitrary robustness against frame dropping, transposition, insertion, decimation, and upsampling. However, increased temporal redundancy has a cost of increased vulnerability to temporal estimation. Spatial synchronization was explored by constructing a watermark in a manner inspired by the work in temporal synchronization. With sufficient spatial redundancy, the detector is able to estimate the rotation and scale of the watermark using the autocorrelation.

In addition to the work in synchronization, a technique was described for authentication watermarking. The semi-fragile watermarking technique is capable of detecting substantial alterations in watermarked images but is also sufficiently insensitive to allow the watermarked image to be compressed using JPEG. Experimental results show that the watermark detector can identify altered blocks with 75% accuracy under moderate compression and with near 90% accuracy under light compression for blocks of  $16 \times 16$  pixels. Independent evaluations have shown that the proposed technique (using larger block sizes) has performance that is comparable with other semi-fragile watermarking techniques described in the literature. Addi-

tional analysis and development can improve the detection performance and security of the proposed technique.

## 6.1 Contributions

- New models for video watermark embedding and detection where the structure of the watermark is expressed as a key sequence produced using a state machine. The watermark detector uses a queue to perform a limited search to establish and maintain synchronization. State machines were described for time-invariant key, time-independent key, and periodic key watermarks. The state machine may represent a pseudo-random number generator as well as many other key generators. The state machine may be defined using cryptographic hash functions.
- The models demonstrate the temporal properties of video watermarks. In particular, the state transition function  $\phi(\cdot)$  induces a chain of states that is produced by the embedder and traced by the detector. The analysis shows that under the proposed framework, watermarks are robust against frame insertion and temporal upsampling attacks. However, frame dropping, decimation, and transposition may disrupt the chain of states, causing desynchronization.
- A method for designing watermarks with demonstrable resilience against frame dropping, transposition, and temporal decimation. By designing temporal redundancy in the watermark, temporal synchronization is lost only after  $\beta$  consecutive frames have been dropped or moved (transposed). Initial synchronization is addressed by resetting the key generator.
- The key sequence produced by the key generator may be video-dependent by the use of a feature extractor, which improves the security of the watermark against estimation, copy, and ambiguity attacks. The state transitions themselves may occur in accordance to the characteristics of the video, which

prevents a loss of temporal redundancy caused by changing feature values and improves the robustness of the watermark.

- A method for designing watermarks with spatial redundancy was proposed, inspired by the temporal synchronization models. A method for template matching was proposed based on the geometric arrangement of the grid of peaks in the autocorrelation of a spatially redundant watermark. Redundancy allows the scale and orientation of the watermark to be estimated under uniform scaling and rotation attacks.
- A semi-fragile watermark for authentication that can detect and localize tampering but is not sensitive to JPEG compression.

## 6.2 Future Work

Synchronization is a significant issue in robust watermark detection and robust watermarking applications such as content protection. The relative ease of performing a synchronization attack and the effectiveness of synchronization attacks to cause detection miss provides the attacker with a tremendous advantage. For example, the watermark may be rendered undetectable by subtle scaling, rotation, or warping. Synchronization is generally a search, which implies an added cost to watermark detection. While we have taken a step towards a fundamental understanding of synchronization, more work is needed. Hopefully, future watermarks shall be more resilient against both spatial and temporal synchronization attacks and more efficient search methods shall be devised to address synchronization issues.

We believe that the approach of modeling watermark construction, embedding, and detection allows a deeper examination of the underlying issues and the design of more effective watermarks. Models should explain the difficulties of current techniques against a particular process (such as synchronization attacks), as well as demonstrate a more effective watermark design. While modeling has been used for resilience against removal attacks, papers discussing synchronization generally de-

scribe or propose methods for creating templates and template matching but often neglect modeling the watermarking process. Modeling the watermarking process is useful in examining the limitations of current techniques before one inserts information (in the form of a template) for synchronization. Despite the differences between temporal and spatial synchronization that was mentioned in this dissertation, a unified model that encompasses both spatial and temporal synchronization would be useful in exploring the limits of synchronization.

There are many synchronization attacks that have not been addressed in this dissertation, such as non-uniform scaling, shearing, cropping, and perspective transformations. Hopefully, models for these attacks and their effects on watermark detection will be devised. Even more difficult attacks include generalized spatiotemporal transformations, particularly those transformations that are neither linear nor spatiotemporally-invariant. New approaches may be needed for addressing these attacks. One potential approach is to examine synchronization as an inverse problem, where the attack obtains  $\hat{Y} = \Psi(Y)$  and the synchronizer attempts to obtain  $\Psi^{-1}$ . Constraints or regularization may be needed to obtain the solution to inverse problems, such as by restricting  $\Phi$  to a conformal mapping. Another potential approach is to extend the state machine concept used in this dissertation towards a Markov-Random Field (MRF) technique to estimate the most likely  $\Psi^{-1}$ , once appropriate boundary conditions are found. The regularization (in the inverse approach) or Markov models (in the MRF approach) may rely on the fact that the coordinate transformation should not damage the perceptual quality of the attacked signal too greatly. For example, synchronization under *any* spatiotemporal warping attack may not be realistic, but synchronization under “subtle” spatiotemporal warping may be achievable.

Additional work in fragile and semi-fragile watermarks will improve the security and detection performance of these watermarks. Many types of security attacks have been identified to produce forgeries, as well as countermeasures against these attacks. Some applications may benefit when authentication allows other types



of information-preserving transformations than lossy compression, such as printing and scanning, contrast adjustment, and slight geometric distortion. Development in semi-fragile watermarks may also devise methods to provide robustness to particular types of information-preserving transformations.

Watermarking is a technology with a promising role in content protection, authentication, and other applications. Vulnerabilities such as detection miss under synchronization attacks demonstrate that additional technical maturity is needed, but perhaps one day the hurdles can be overcome.

## LIST OF REFERENCES

## LIST OF REFERENCES

- [1] A. M. Eskicioglu and E. J. Delp, “An overview of multimedia content protection in consumer electronics devices,” *Signal Processing: Image Communication*, vol. 16, no. 7, pp. 681–699, Apr. 2001.
- [2] E. T. Lin, A. M. Eskicioglu, R. L. Lagendijk, and E. J. Delp, “Advances in digital video content protection,” *Proceedings of the IEEE: Special Issue on Advances in Video Coding and Delivery*, vol. 93, no. 1, pp. 171–183, Jan. 2005.
- [3] F. Hartung, “Digital watermarking and fingerprinting of uncompressed and compressed video,” Ph.D. dissertation, Universität Erlangen–Nürnberg, 1999.
- [4] Kazaa software version 2.6. <http://www.kazaa.com>
- [5] B. Cohen. (2001) BitTorrent. <http://bitconjurer.org/BitTorrent/index.html>
- [6] eDonkey and Overnet. <http://www.edonkey2000.com>
- [7] P. Biddle, P. England, M. Peinado, and B. Willman, “The Darknet and the future of content distribution,” *Proceedings of the ACM Workshop on Digital Rights Management*, Washington D.C., Nov. 18, 2002. <http://crypto.stanford.edu/DRM2002/darknet5.doc>
- [8] R. Parloff, “Morpheus falling?” *IEEE Spectrum*, vol. 40, no. 12, pp. 18–19, Dec. 2003.
- [9] E. W. Felten, “A skeptical view of DRM and fair use,” *Communications of the ACM: Special Issue on Digital Rights Management and Fair Use by Design*, vol. 46, no. 4, pp. 56–59, Apr. 2003.
- [10] P. Samuelson, “DRM {And, Or, Vs.} the law,” *Communications of the ACM: Special Issue on Digital Rights Management and Fair Use by Design*, vol. 46, no. 4, pp. 41–45, Apr. 2003.
- [11] J. Litman, *Digital Copyright*. Amherst, NY: Prometheus Books, 2001.
- [12] D. K. Mulligan, J. Han, and A. J. Burstein, “How DRM-based content delivery systems disrupt expectations of “personal use”,” *Proceedings of the ACM workshop on Digital Rights Management*, Washington D.C., 2003, pp. 77–89.
- [13] D. Clark, “How copyright became controversial,” *Proceedings of the 12th Annual Conference on Computers, Freedom and Privacy*, San Francisco, CA, 2002.

- [14] J. A. Bloom, I. J. Cox, T. Kalker, J.-P. M. G. Linnartz, M. L. Miller, and C. B. S. Traw, "Copy protection for dvd video," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1267–1276, July 1999.
- [15] E. Diehl and T. Furon, "©watermark: Closing the analog hole," *Proceedings of the IEEE International Conference on Consumer Electronics*, 2003, pp. 52–53.
- [16] B. Schneier, *Applied Cryptography*. New York, NY: John Wiley and Sons, Inc, 1996.
- [17] D. R. Stinson, *Cryptography Theory and Practice*. Boca Raton, FL: CRC Press, 1995.
- [18] H. Beker and F. Piper, *Cipher Systems: The Protection of Communications*. New York, NY: John Wiley & Sons, Inc., 1982.
- [19] J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.
- [20] X. Liu and A. M. Eskicioglu, "Selective encryption of multimedia content in distribution networks: Challenges and new directions," *Proceedings of the 2nd International Conference on Communications, Internet, and Information Technology*, Scottsdale, AZ, Nov.17–19, 2003.
- [21] N. Bourbakis and A. Dollas, "SCAN-based compression-encryption-hiding for video on demand," *IEEE Transactions on Multimedia*, pp. 79–87, July–Sept. 2003.
- [22] W. Zeng and S. Lei, "Efficient frequency domain selective scrambling of digital video," *IEEE Transactions on Multimedia*, vol. 5, no. 1, pp. 118–129, Mar. 2003.
- [23] J. G. Wen, M. Severa, W. Zeng, M. H. Luttrell, and W. Jin, "A format-compliant configurable encryption framework for access control of video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 545–557, June 2002.
- [24] *Announcing the Advanced Encryption Standard (AES)*, National Institute of Standards and Technology Std. FIPS-197, Nov. 2001. <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- [25] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," *IEEE Transactions on Signal Processing: Supplement on Secure Media*, vol. 52, no. 10, pp. 2992–3006, Oct. 2004.
- [26] M. Barni and F. Bartolini, "Data hiding for fighting piracy," *IEEE Signal Processing Magazine*, vol. 21, no. 2, pp. 28–39, Mar. 2004.
- [27] G. Doërr and J.-L. Dugelay, "A guide tour of video watermarking," *Signal Processing: Image Communication*, vol. 18, no. 4, pp. 263–282, Apr. 2003.
- [28] I. Cox, M. Miller, and J. Bloom, *Digital Watermarking*. San Francisco, CA: Morgan Kaufmann, 2002.

- [29] C. I. Podilchuk and E. J. Delp, "Digital watermarking: Algorithms and applications," *IEEE Signal Processing Magazine*, vol. 18, no. 4, pp. 33–46, July 2001.
- [30] G. Langelaar, I. Setyawan, and R. Lagendijk, "Watermarking digital image and video data: A state-of-the-art overview," *IEEE Signal Processing Magazine*, vol. 17, no. 5, pp. 20–46, Sept. 2000.
- [31] F. Hartung and M. Kutter, "Multimedia watermarking techniques," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1079–1107, July 1999.
- [32] R. Wolfgang, C. Podilchuk, and E. Delp, "Perceptual watermarks for digital images and video," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1108–1126, July 1999.
- [33] M. Swanson, M. Kobayashi, and A. Tewfik, "Multimedia data-embedding and watermarking technologies," *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1064–1087, June 1998.
- [34] C. de Vleeschouwer, J.-F. Delaigle, and B. Macq, "Invisibility and application functionalities in perceptual watermarking—an overview," *Proceedings of the IEEE*, vol. 90, no. 1, pp. 64–77, Jan. 2002.
- [35] I. Cox, M. Miller, and J. Bloom, "Watermarking applications and their properties," *Proceedings of the International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, Mar. 27–29, 2000, pp. 6–10.
- [36] G. C. Langelaar, "Real-time watermarking techniques for compressed video data," Ph.D. dissertation, Delft University of Technology, 2000.
- [37] R. J. Anderson and F. A. Petitcolas, "On the limits of steganography," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 474–482, May 1998.
- [38] T. Kalker, G. Depovere, J. Haitsma, and M. Maes, "A video watermarking system for broadcast monitoring," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents*, vol. 3657, San Jose, CA, Jan. 25–27, 1999, pp. 103–112.
- [39] F. Bartolini, A. Tefas, M. Barni, and I. Pitas, "Image authentication techniques for surveillance applications," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1403–1418, Oct. 2001.
- [40] G. Friedman, "The trustworthy digital camera: Restoring credibility to the photographic image," *IEEE Transactions on Consumer Electronics*, vol. 39, pp. 905–910, Nov. 1993.
- [41] P. W. Wong and N. Memon, "Secret and public key image watermarking schemes for image authentication and ownership verification," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1593–1601, Oct. 2001.
- [42] M. U. Celik, G. Sharma, E. Saber, and A. M. Tekalp, "Hierarchical watermarking for secure image authentication with localization," *IEEE Transactions on Image Processing*, vol. 11, no. 6, pp. 585–595, June 2002.

- [43] E. T. Lin, C. I. Podilchuk, and E. J. Delp, "Detection of image alterations using semi-fragile watermarks," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 23–28, 2000, pp. 152–163.
- [44] E. T. Lin and E. J. Delp, "A review of fragile image watermarks," *Proceedings of the Multimedia and Security Workshop at ACM Multimedia '99*, Orlando, FL, Oct. 30–31, 1999, pp. 25–29.
- [45] J. Picard, C. Vielhauer, and N. Thorwirth, "Towards fraud-proof ID documents using multiple data hiding technologies and biometrics," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, San Jose, CA, Jan. 19–22, 2004, pp. 416–427.
- [46] A. M. Alattar, "'Smart images' using Digimarc's watermarking technology," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 24–26, 2000, pp. 264–273.
- [47] S. Katzenbeisser and F. A. P. Petitcolas, Eds., *Information Hiding: Techniques for Steganography and Digital Watermarking*. Norwood, MA: Artech House, 2000.
- [48] B. A. Wandell, *Foundations of Vision*. Sunderland, MA: Sinauer Associates, Inc., 1995.
- [49] R. N. Haber, Ed., *Information-Processing Approaches to Visual Perception*. New York, NY: Holt, Rinehart, and Winston, Inc., 1969.
- [50] A. Bovik, Ed., *Handbook of Image & Video Processing*. San Diego, CA: Academic Press, 2000.
- [51] N. Jayant, J. Johnston, and R. Safranek, "Signal compression based on models of human perception," *Proceedings of the IEEE*, vol. 81, no. 10, pp. 1385–1422, Oct. 1993.
- [52] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.
- [53] D. Pan, "A tutorial on MPEG/audio compression," *IEEE Transactions on Multimedia*, vol. 2, no. 2, pp. 60–74, Summer 1995.
- [54] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.
- [55] W. Fumy and P. Landrock, "Principles of key management," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 5, pp. 785–793, June 1993.
- [56] J. J. Eggers, R. Bäuml, R. Tzschoppe, and B. Girod, "Scalar costa scheme for information embedding," *IEEE Transactions on Signal Processing*, vol. 15, no. 4, pp. 1003–1019, Apr. 2003.
- [57] J. K. Su and B. Girod, "Power-spectrum condition for energy-efficient watermarking," *IEEE Transactions on Multimedia*, vol. 4, no. 4, pp. 551–560, Dec. 2002.

- [58] M. L. Miller, G. Doërr, and I. J. Cox, "Applying informed coding and embedding to design a robust high-capacity watermark," *IEEE Transactions on Image Processing*, vol. 13, no. 6, pp. 792–807, June 2004.
- [59] J. Eggers and B. Girod, *Informed Watermarking*. Boston, MA: Kluwer Academic Publishers, 2002.
- [60] M. Barni, F. Bartolini, and A. D. Rosa, "Advantages and drawbacks of multiplicative spread spectrum watermarking," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents V*, Santa Clara, CA, Jan. 21–24, 2003, pp. 290–299.
- [61] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoan, "Secure spread spectrum watermarking for multimedia," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1673–1687, Dec. 1997.
- [62] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer Academic Publishers, 1992.
- [63] B. Chen and G. W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1423–1443, May 2001.
- [64] J. Oostveen, T. Kalker, and M. Staring, "Adaptive quantization watermarking," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan. 19–24, 2004, pp. 296–303.
- [65] R. Bäuml, R. Tzschoppe, A. Kaup, and J. Huber, "Optimality of SCS watermarking," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, CA, Jan. 21–24, 2003, pp. 612–622.
- [66] K. R. Rao and P. Yip, *Discrete Cosine Transform - Algorithms, Advantages, Applications*. New York, NY: Academic Press, 1990.
- [67] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. 23, no. 23, pp. 90–93, Jan. 1974.
- [68] S. Voloshynovskiy, S. Pereira, V. Iquise, and T. Pun, "Attack modelling: Towards a second generation watermarking benchmark," *Signal Processing*, vol. 81, no. 6, pp. 1177–1214, June 2001.
- [69] S. Voloshynovskiy, S. Pereira, T. Pun, J. J. Eggers, and J. K. Su, "Attacks on digital watermarks: Classification, estimation-based attacks, and benchmarks," *IEEE Communications Magazine*, vol. 39, no. 8, pp. 118–126, Aug. 2001.
- [70] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Attacks on copyright marking systems," *Proceedings of the Second International Workshop on Information Hiding*, Portland, OR, Apr. 15–17, 1998, pp. 219–239.
- [71] J. J. Eggers and B. Girod, "Quantization effects on digital watermarks," *Signal Processing*, vol. 81, no. 2, pp. 239–263, Feb. 2001.

- [72] J. Xin, C.-W. Lin, and M.-T. Sun, "Digital video transcoding," *Proceedings of the IEEE: Special Issue on Advances in Video Coding and Delivery*, vol. 93, no. 1, pp. 84–97, Jan. 2005.
- [73] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: An overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18–29, Mar. 2003.
- [74] I. D. Shterev, I. L. Lagendijk, and R. Heusdens, "Statistical amplitude scale estimation for quantization-based watermarking," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, Jan.19–22, 2004, pp. 796–804.
- [75] D. Kirovski and F. A. P. Petitcolas, "Blind pattern matching attack on watermarking systems," *IEEE Transactions on Signal Processing*, vol. 51, no. 4, pp. 1045–1053, Apr. 2003.
- [76] M. F. Mansour and A. H. Tewfik, "Attacks on quantization-based watermarking schemes," *Proceedings of the IEEE Seventh International Symposium on Signal Processing and its Applications*, vol. 2, July 1–4, 2003, pp. 367–370.
- [77] S. A. Craver, M. Wu, B. Liu, A. Stubblefield, B. Swartzlander, D. W. Wallach, D. Dean, and E. W. Felten, "Reading between the lines: Lessons from the SDMI challenge," *Proceedings of the 10th USENIX Security Symposium*, Washington, D.C., Aug. 13–17 2001.
- [78] S. Voloshynovskiy, S. Pereira, A. Herrigel, N. Baumgartner, and T. Pun, "Generalized watermarking attack based on watermark estimation and perceptual remodulation," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 24–26, 2000, pp. 358–370.
- [79] R. Barnett and D. E. Pearson, "Frequency Mode LR attack operator for digitally watermarked images," *Electronics Letters*, vol. 19, pp. 1837–1838, 1998.
- [80] G. C. Langelaar, R. L. Lagendijk, and J. Biemond, "Removing spatial spread spectrum watermarks," *Proceedings of the European Signal Processing Conference (EUSIPCO'98)*, Rhodes, Greece, Sept. 8–11, 1998.
- [81] I. J. Cox and J.-P. M. G. Linnartz, "Some general methods for tampering with watermarks," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 587–593, May 1998.
- [82] G. Doërr and J.-L. Dugelay, "Security pitfalls of frame-by-frame approaches to video watermarking," *IEEE Transactions on Signal Processing: Supplement on Secure Media*, vol. 52, no. 10, pp. 2955–2964, Oct. 2004.
- [83] —, "Switching between orthogonal watermarks for enhanced security against collusion in video," Eurécom Institute, Tech. Rep. RR-03-080, July 4, 2003. <http://www.eurecom.fr/~doerr>
- [84] M. Wu, W. Trappe, Z. J. Wang, and K. J. R. Liu, "Collusion resistant multimedia fingerprinting: A unified framework," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan. 19–22, 2004, pp. 748–759.



- [85] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Transactions on Information Theory*, vol. 44, no. 5, pp. 1897–1905, Sept. 1998.
- [86] M. U. Celik, G. Sharma, and A. M. Tekalp, "Collusion-resilient fingerprinting using random pre-warping," *Proceedings of the IEEE International Conference on Image Processing*, Sept. 14–17, 2003, pp. 509–512.
- [87] W. Trappe, M. Wu, and K. J. R. Liu, "Anti-collusion codes: Multi-user and multimedia perspectives," *Proceedings of the IEEE International Conference on Image Processing*, vol. 2, Sept. 22–25, 2002, pp. 149–152.
- [88] M. Kutter, S. Voloshynovskiy, and A. Herrigel, "The watermark copy attack," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 24–26, 2000, pp. 371–380.
- [89] M. Holliman, W. Macy, and M. M. Yeung, "Robust frame-dependent video watermarking," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 24–26, 2000, pp. 186–197.
- [90] F. Deguillaume, S. Voloshynovskiy, and T. Pun, "Secure hybrid robust watermarking resistant against tampering and copy attack," *Signal Processing*, vol. 83, no. 10, pp. 2133–2170, Oct. 2003.
- [91] C.-S. Lu, H.-Y. M. Liao, and M. Kutter, "Denoising and copy attacks resilient watermarking by exploiting prior knowledge at detector," *IEEE Transactions on Image Processing*, vol. 11, no. 3, pp. 280–292, Mar. 2002.
- [92] A. Adelsbach, S. Katzenbeisser, and H. Veith, "Watermarking schemes provably secure against copy and ambiguity attacks," *Proceedings of the ACM Workshop on Digital Rights Management*, Washington D.C., 2003, pp. 111–119.
- [93] S. Craver, N. Memon, B.-L. Yeo, and M. M. Yeung, "Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 573–586, May 1998.
- [94] W. Zeng and B. Liu, "A statistical watermark detection technique without using original images for resolving rightful ownerships of digital images," *IEEE Transactions on Image Processing*, vol. 8, no. 11, pp. 1534–1548, Nov. 1999.
- [95] S. Craver, "The return of ambiguity attacks," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 252–259.
- [96] R. Liu and T. Tan, "An SVD-based watermarking scheme for protecting rightful ownership," *IEEE Transactions on Multimedia*, vol. 4, no. 1, pp. 121–128, Mar. 2002.
- [97] J. Picard and A. Robert, "On the public key watermarking issue," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents III*, San Jose, CA, Jan. 22–25, 2001, pp. 290–299.
- [98] F. Hartung and B. Girod, "Fast public-key watermarking of compressed video," *Proceedings of the IEEE International Conference on Image Processing 1997*, vol. 1, Santa Barbara, CA, Oct. 26–29, 1997, pp. 528–531.

- [99] H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Part I*. New York, NY: John Wiley & Sons, Inc., 2001.
- [100] G. Casella and R. L. Berger, *Statistical Inference*. Belmont, CA: Duxbury Press, 1990.
- [101] M. D. Srinath, P. K. Rajasekaram, and R. Viswanathan, *Introduction to Statistical Signal Processing with Applications*. Englewood Cliffs, NJ: Prentice Hall, 1996.
- [102] W. B. Davenport, Jr. and W. L. Root, *An Introduction to the Theory of Random Signals and Noise*. New York, NY: IEEE Press, 1987.
- [103] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York, NY: McGraw-Hill, Inc., 1991.
- [104] G. Langelaar and R. Lagendijk, "Optimal differential energy watermarking of DCT encoded images and video," *IEEE Transactions on Image Processing*, vol. 10, no. 1, pp. 148–158, Jan. 2001.
- [105] M. Barni, F. Bartolini, A. D. Rosa, and A. Piva, "A new decoder for the optimum recovery of nonadditive watermarks," *IEEE Transactions on Image Processing*, vol. 10, no. 5, pp. 755–766, May 2001.
- [106] A. S. Cohen and A. Lapidoth, "The gaussian watermarking game," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1639–1667, June 2002.
- [107] C. B. Peel, "On 'dirty-paper coding'," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 112–113, May 2003.
- [108] M. H. M. Costa, "Writing on dirty paper," *IEEE Transactions on Information Theory*, vol. IT-29, no. 3, pp. 439–441, May 1983.
- [109] G. Depovere, T. Kalker, and J.-P. Linnartz, "Improved watermark detection reliability using filtering before correlation," *Proceedings of the IEEE International Conference on Image Processing 1998*, vol. 1, Chicago, IL, Oct. 4–7, 1998, pp. 430–434.
- [110] F. A. P. Petitcolas, "Watermarking schemes evaluation," *IEEE Signal Processing Magazine*, vol. 17, no. 5, pp. 58–64, Sept. 2000.
- [111] F. Mintzer, G. W. Braudaway, and M. M. Yeung, "Effective and ineffective digital watermarks," *Proceedings of the IEEE International Conference on Image Processing 1997*, Santa Barbara, CA, Oct. 1997, pp. 9–12.
- [112] A. B. Watson, J. Hu, and J. F. McGowan III, "Digital video quality metric based on human vision," *Journal of Electronic Imaging*, vol. 10, no. 1, pp. 20–29, 2001.
- [113] A. N. Netravali and B. G. Haskell, *Digital Pictures Representation and Compression*. New York, NY: Plenum Press, 1998.
- [114] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

- [115] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 121–132, Feb. 2004.
- [116] M. A. Masry and S. S. Hemami, "A metric for continuous quality evaluation of compressed video with severe distortions," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 133–146, Feb. 2004.
- [117] S. J. P. Westen, R. L. Lagendijk, and J. Biemond, "Spatio-temporal model of human vision for digital video compression," *Proceedings of the SPIE Human Vision and Electronic Imaging II*, vol. 3016, San Jose, CA, 1997, pp. 260–268.
- [118] C. J. van den Branden Lambrecht and O. Verscheure, "Perceptual quality measure using a spatio-temporal model of the human visual system," *Proceedings of the SPIE Conference on Digital Video Compression: Algorithms and Technologies*, vol. 2668, San Jose, CA, Jan./Feb. 1996, pp. 450–461.
- [119] A. Basso, İ. Dalgıç, F. A. Tobagi, and C. J. van den Branden Lambrecht, "Study of MPEG–2 coding performance based on a perceptual quality metric," *Proceedings of the 1996 Picture Coding Symposium*, Melbourne, Australia, Mar. 1996, pp. 263–268.
- [120] A. A. Webster, C. T. Jones, M. H. Pinson, S. D. Voran, and S. Wolf, "An objective video quality assessment system based on human perception," *Proceedings of the SPIE Human Vision, Visual Processing, and Digital Displays IV*, San Jose, CA, Feb. 1993, pp. 15–26.
- [121] F. X. J. Lukas and Z. L. Budrikis, "Picture quality prediction based on a visual model," *IEEE Transactions on Communications*, vol. COM-30, no. 7, pp. 1679–1692, July 1982.
- [122] M. Wu and B. Liu, "Data hiding in image and video: Part I—fundamental issues and solutions," *IEEE Transactions on Image Processing*, vol. 12, no. 6, pp. 685–695, June 2003.
- [123] P. Moulin and J. A. O'Sullivan, "Information-theoretic analysis of information hiding," *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 563–593, Mar. 2003.
- [124] M. Barni, F. Bartolini, and T. Furon, "A general framework for robust watermark security," *Signal Processing*, vol. 83, no. 10, pp. 2069–2084, Oct. 2003.
- [125] H. C. Kim, H. Ogunley, O. Guitart, and E. J. Delp, "The watermark evaluation testbed," *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking VI*, vol. 5306, San Jose, CA, Jan. 19–22, 2004, pp. 236–247.
- [126] B. Macq, J. Dittmann, and E. J. Delp, "Benchmarking of image watermarking algorithms for digital rights management," *Proceedings of the IEEE*, vol. 92, no. 6, pp. 971–984, June 2004.
- [127] J. C. Vorbrüggen and F. Cayre, "The Certimark benchmark: Architecture and future perspectives," *Proceedings of the IEEE International Conference on Multimedia and Expo 2002*, vol. 2, Lausanne, Switzerland, Aug. 26–29, 2002, pp. 485–488.

- [128] S. Pereira, S. Voloshynovskiy, M. Madueno, S. Marchand-Maillet, and T. Pun, "Second generation benchmarking and application oriented evaluation," *Information Hiding Workshop*, Pittsburgh, PA, Apr. 2001.
- [129] M. Steinebach, F. A. P. Petitcolas, F. Raynal, J. Dittmann, C. Fontaine, C. Seibel, N. Fatès, and L. C. Ferri, "StirMark benchmark: Audio watermarking attacks," *Proceedings of the International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, Apr. 2–4, 2001, pp. 49–54.
- [130] V. Solachidis, A. Tefas, N. Nikolaidis, S. Tsekeridou, A. Nikolaidis, and I. Pitas, "A benchmarking protocol for watermarking methods," *Proceedings of the IEEE International Conference on Image Processing 2001*, vol. 3, Thessaloniki, Greece, Oct. 7–10, 2001, pp. 1023–1026.
- [131] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Englewood Cliffs, NJ: Prentice Hall, 1995.
- [132] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY: John Wiley & Sons, Inc., 1991.
- [133] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, Nov. 1998.
- [134] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.
- [135] J. S. Lim, *Two-Dimensional Signal and Image Processing*. Englewood Cliffs, NJ: Prentice Hall, 1990.
- [136] C. A. Poynton, *A Technical Introduction to Digital Video*. New York, NY: John Wiley & Sons, Inc., 1996.
- [137] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 1996.
- [138] A. V. Oppenheim, A. S. Willsky, and I. T. Young, *Signals and Systems*. Englewood Cliffs, NJ: Prentice Hall, 1983.
- [139] K. Jack, *Video Demystified*. San Diego, CA: HighText Publications, 1996.
- [140] D. H. Hubel, *Eye, Brain, and Vision*. New York, NY: Scientific American Library, 1995.
- [141] R. M. Boynton, *Human Color Vision*. USA: Optical Society of America, 1992.
- [142] G. Sharma and H. J. Trussell, "Digital color imaging," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 901–932, July 1997.
- [143] S. Süsstrunk, R. Buckley, and S. Swen, "Standard RGB color spaces," *Proceedings of the IS&T/SID Seventh Color Imaging Conference: Color Science, Systems and Applications*, Scottsdale, AZ, Nov. 1999, pp. 127–134.

- [144] *Parameter Values for the HDTV Standards for Production and International Programme Exchange*, International Telecommunication Union Std. ITU-R BT.709-3, 1998.
- [145] *Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios*, International Telecommunication Union Std. ITU-R BT.601-5, 1995.
- [146] *Television – 1920 × 1080 Scanning and Analog and Parallel Digital Interfaces for Multiple Picture Rates*, Society of Motion Picture and Television Engineers Std. 274M, 2003.
- [147] *Television – 1280 × 720 Progressive Image Sample Structure – Analog and Digital Representation and Analog Interface*, Society of Motion Picture and Television Engineers Std. 296M, 2001.
- [148] J. Taylor, *DVD Demystified*. New York, NY: McGraw-Hill, 1998.
- [149] E. T. Lin, C. I. Podilchuk, T. Kalker, and E. J. Delp, “Streaming video and rate scalable compression: What are the challenges for watermarking?” *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 198–205, Jan. 2004.
- [150] X. Li, M. H. Ammar, and S. Paul, “Video multicast over the internet,” *IEEE Network*, vol. 13, no. 2, pp. 46–60, Mar.–Apr. 1999.
- [151] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. M. Peha, “Streaming video over the internet: Approaches and directions,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–300, Mar. 2001.
- [152] Advanced television systems committee. <http://www.atsc.org>
- [153] A. Tekalp, *Digital Video Processing*. Upper Saddle River, NJ: Prentice Hall, 1995.
- [154] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Boston, MA: Kluwer Academic Publishers, 1997.
- [155] L. Torres and M. Kunt, Eds., *Video Coding: The Second Generation Approach*. Boston, MA: Kluwer Academic Publishers, 1996.
- [156] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*. New York, NY: Chapman & Hall, 1997.
- [157] J.-R. Ohm, “Advances in scalable video coding,” *Proceedings of the IEEE: Special Issue on Advances in Video Coding and Delivery*, vol. 93, no. 1, pp. 42–56, Jan. 2005.
- [158] W. Li, “Overview of fine granularity scalability in MPEG-4 video standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, Mar. 2001.
- [159] K. Shen and E. J. Delp, “Wavelet based rate scalable video compression,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 1, pp. 109–122, Feb. 1999.

- [160] H.-C. Huang, C.-N. Wang, and T. Chiang, "A robust fine granularity scalability using trellis-based predictive leak," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 372–385, June 2002.
- [161] Y. Liu, Z. Li, P. Salama, and E. J. Delp, "A discussion of leaky prediction based scalable coding," *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, vol. 2, Baltimore, MD, July 6–9, 2003, pp. 565–568.
- [162] Y. Liu, "Layered scalable and low complexity video encoding: New approaches and theoretic analysis," Ph.D. dissertation, Purdue University, Aug. 2004.
- [163] M. Karczewicz and R. Kurceren, "The SP- and SI-frames design for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 637–644, July 2003.
- [164] R. Talluri, "Error-resilient video coding in the ISO MPEG-4 standard," *IEEE Communications Magazine*, vol. 36, no. 6, pp. 112–119, June 1998.
- [165] Y. Wang, S. Wenger, J. Wen, and A. K. Katsaggelos, "Error resilient video coding techniques," *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61–82, July 2000.
- [166] Y. Wang and Q.-F. Zhu, "Error control and concealment for video communications: A review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974–997, May 1998.
- [167] *Information Technology – Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbit/s*, International Organization for Standardization Std. ISO/IEC 11 172-2, May 1993.
- [168] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, Eds., *MPEG Video Compression Standard*. New York, NY: Chapman & Hall, 1997.
- [169] *Line transmission of Non-Telephone Signals: Video Codec for Audiovisual Services at  $p \times 64$  kbits*, International Telecommunication Union Std. ITU-T H.261, Mar. 1993.
- [170] *Multiplexing Protocol for Low Bit Rate Multimedia Communication*, International Telecommunication Union Std. ITU-T H.223, July 2001.
- [171] *Information Technology – Generic Coding of Moving Pictures and Associated Audio Information*, International Organization for Standardization Std. ISO/IEC 13 818-2, 1994.
- [172] *Coding for Low Bitrate Communication*, International Telecommunication Union Std. ITU-T H.263, Mar. 1996.
- [173] G. Côté, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video coding at low bit rates," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 849–866, Nov. 1998.
- [174] *Information Technology – Coding of Audio-Visual Objects: Video*, International Organization for Standardization Std. ISO/IEC 14 496-2, Oct. 1998.

- [175] T. Ebrahimi and C. Horne, "MPEG-4 natural video coding—an overview," *Signal Processing: Image Communication*, vol. 15, no. 4, pp. 365–385, Jan. 2000.
- [176] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 598–603, July 2003.
- [177] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 614–619, July 2003.
- [178] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, July 2003.
- [179] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 688–703, July 2003.
- [180] Y. Liu, P. Salama, G. W. Cook, and E. J. Delp, "Rate-distortion analysis of layered video coding by leaky prediction," *Proceedings of the SPIE International Conference on Video Communications and Image Processing (VCIP)*, vol. 5308, San Jose, CA, Jan. 18–22, 2004, pp. 543–554.
- [181] H. Feng and M. Effros, "On the rate-distortion performance and computational efficiency of the Karhunen-Loève Transform for lossy data compression," *IEEE Transactions on Image Processing*, vol. 11, no. 2, pp. 113–122, Feb. 2002.
- [182] K. I. Diamantaras and M. G. Strintzis, "Optimal transform coding in the presence of quantization noise," *IEEE Transactions on Image Processing*, vol. 8, no. 11, pp. 1508–1515, Nov. 1999.
- [183] V. R. Algazi and D. J. Sakrison, "On the optimality of the Karhunen-Loève expansion," *IEEE Transactions on Information Theory*, vol. IT-15, no. 2, pp. 319–321, Mar. 1969.
- [184] M. Vetterli, "Fast 2-D discrete cosine transform," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '85)*, vol. 10, Apr. 1985, pp. 1538–1541.
- [185] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.
- [186] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, July 2000.
- [187] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, Apr. 1992.

- [188] B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proceedings of the IEEE: Special Issue on Advances in Video Coding and Delivery*, vol. 93, no. 1, pp. 71–83, Jan. 2005.
- [189] R. B. Wolfgang and E. J. Delp, "A watermark for digital images," *Proceedings of the IEEE International Conference on Image Processing 1996*, vol. 3, Lausanne, Switzerland, Sept. 16–19, 1996, pp. 219–222.
- [190] —, "Fragile watermarking using the VW2D watermark," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents I*, vol. 3657, San Jose, CA, Jan. 25–27, 1999, pp. 204–213.
- [191] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne, "A digital watermark," *Proceedings of the IEEE International Conference on Image Processing 1994*, vol. 2, Austin, TX, Nov. 13–16, 1994, pp. 86–90.
- [192] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," *IBM Systems Journal*, vol. 35, no. 3 & 4, pp. 313–336, 1996. <http://www.almaden.ibm.com/cs/people/dgruhl/313.pdf>
- [193] G. C. Langelaar, J. van der Lubbe, and J. Biemond, "Copy protection for multimedia data based on labeling techniques," *Proceedings of the 17th Symposium on Information Theory in the Benelux*, Enschede, May 1996, pp. 33–39.
- [194] W. Zeng and B. Liu, "On resolving rightful ownerships of digital images by invisible watermarks," *Proceedings of the IEEE International Conference on Image Processing 1997*, vol. 1, Santa Barbara, CA, Oct. 26–29, 1997, pp. 552–555.
- [195] S. Walton, "Information authentication for a slippery new age," *Dr. Dobbs Journal*, vol. 20, no. 4, pp. 18–26, Apr. 1995.
- [196] G. Caronni, "Assuring ownership rights for digital images," *Proceedings of Reliable IT Systems VIS '95*, H. H. Brüggemann and W. Gerhardt-Häckl, Eds., 1995. <http://www.olymp.org/~caronni/work/papers/givis-final.pdf>
- [197] M. Kutter, F. Jordan, and F. Bossen, "Digital watermarking of color images using amplitude modulation," *Journal of Electronic Imaging*, vol. 7, no. 2, pp. 326–332, Apr. 1998.
- [198] M. Kutter and S. Winkler, "A vision-based masking model for spread-spectrum image watermarking," *IEEE Transactions on Image Processing*, vol. 11, no. 1, pp. 16–25, Jan. 2002.
- [199] M. Barni, F. Bartolini, and A. Piva, "Improved wavelet-based watermarking through pixel-wise masking," *IEEE Transactions on Image Processing*, vol. 10, no. 5, pp. 783–791, May 2001.
- [200] Y. Liu, B. Ni, X. Feng, and E. J. Delp, "LOT-based adaptive image watermarking," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, San Jose, CA, Jan. 19–22, 2004, pp. 513–523.
- [201] J. F. Delaigle, C. D. Vleeschouwer, and B. Macq, "Watermarking algorithm based on a human visual model," *Signal Processing*, vol. 66, no. 3, pp. 337–355, May 1998.



- [202] C. I. Podilchuk and W. Zeng, "Image-adaptive watermarking using visual models," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 525–539, May 1998.
- [203] M. D. Swanson, B. Zhu, and A. H. Tewfik, "Transparent robust image watermarking," *Proceedings of the IEEE International Conference on Image Processing 1996*, vol. 3, Lausanne, Switzerland, Sept. 16–19, 1996, pp. 211–214.
- [204] A. B. Watson, "DCT quantization matrices visually optimized for individual images," *Proceedings of the SPIE Conference on Human Vision, Visual Processing, and Digital Display IV*, B. E. Rogowitz, Ed., 1993, pp. 202–216. <http://vision.arc.nasa.gov/publications/spie93abw/spie93abw.pdf>
- [205] A. Robert and J. Picard, "Masking models and watermark unDetection," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents III*, vol. 4314, San Jose, CA, Jan. 22–25, 2001, pp. 455–467.
- [206] G. W. Braudaway, "Protecting publicly-available images with an invisible image watermark," *Proceedings of the IEEE International Conference on Image Processing 1997*, vol. 1, Santa Barbara, CA, Oct. 26–29, 1997, pp. 524–527.
- [207] C.-T. Hsu and J.-L. Wu, "Hidden signatures in images," *Proceedings of the IEEE International Conference on Image Processing 1996*, vol. 3, Sept. 16–19, 1996, pp. 223–226.
- [208] S. Tsekeridou and I. Pitas, "Wavelet-based self-similar watermarking for still images," *Proceedings of the 2000 IEEE International Symposium on Circuits and Systems*, vol. 1, Geneva, Switzerland, May28–31, 2000, pp. 220–223.
- [209] S. Pereira and T. Pun, "Robust template matching for affine resistant image watermarks," *IEEE Transactions on Image Processing*, vol. 9, no. 6, pp. 1123–1129, June 2000.
- [210] V. Solachidis and I. Pitas, "Circularly symmetric watermark embedding in the 2-D DFT domain," *IEEE Transactions on Image Processing*, vol. 10, no. 11, pp. 1741–1753, Nov. 2001.
- [211] J. K. Su, J. J. Eggers, and B. Girod, "Analysis of digital watermarks subjected to optimum linear filtering and additive noise," *Signal Processing*, vol. 81, no. 6, pp. 1141–1175, June 2001.
- [212] J. Dittmann, T. Fiebig, and R. Steinmetz, "A new approach for transformation invariant image and video watermarking in the spatial domain: SSP—self spanning patterns," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 24–26, 2000, pp. 176–185.
- [213] A. Nikolaidis and I. Pitas, "Region-based image watermarking," *IEEE Transactions on Image Processing*, vol. 10, no. 11, pp. 1726–1740, Nov. 2001.
- [214] M. Barni, F. Bartolini, and A. Piva, "Multichannel watermarking of color images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 3, pp. 142–156, Mar. 2002.

- [215] X. Kang, J. Huang, Y. Q. Shi, and Y. Lin, "A DWT-DFT composite watermarking scheme robust to both affine transform and JPEG compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 776–786, Aug. 2003.
- [216] X.-G. Xia, C. G. Boncelet, and G. R. Arce, "A multiresolution watermark for digital images," *Proceedings of the IEEE International Conference on Image Processing 1997*, vol. 1, Santa Barbara, CA, Oct. 26–29, 1997, pp. 548–551.
- [217] W. Zhu, Z. Xiong, and Y.-Q. Zhang, "Multiresolution watermarking for images and video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 545–550, June 1999.
- [218] R. Dugad, K. Ratakonda, and N. Ahuja, "A new wavelet-based scheme for watermarking images," *Proceedings of the IEEE International Conference on Image Processing 1998*, vol. 2, Chicago, IL, Oct. 4–7, 1998, pp. 419–423.
- [219] H. Inoue, A. Miyazaki, and T. Katsura, "An image watermarking method based on the wavelet transform," *Proceedings of the IEEE International Conference on Image Processing 1999*, vol. 1, Kobe, Japan, Oct. 24–28, 1999, pp. 296–300.
- [220] D. Kundur and D. Hatzinakos, "Digital watermarking for telltale tamper proofing and authentication," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1167–1180, July 1999.
- [221] P. Meerwald and A. Uhl, "A survey of wavelet-domain watermarking algorithms," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents III*, vol. 4314, San Jose, CA, Jan. 22–25, 2001, pp. 505–516.
- [222] S. Kang and Y. Aoki, "Digital image watermarking by Fresnel Transform and its robustness," *Proceedings of the IEEE International Conference on Image Processing 1999*, vol. 2, Kobe, Japan, Oct. 24–28, 1999, pp. 221–225.
- [223] C.-Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, M. L. Miller, and Y. M. Lui, "Rotation, scale, and translation resilient watermarking for images," *IEEE Transactions on Image Processing*, vol. 10, no. 5, pp. 767–782, May 2001.
- [224] J. J. K. Ó Ruanaidh and T. Pun, "Rotation, scale and translation invariant spread spectrum digital image watermarking," *Signal Processing*, vol. 66, no. 3, pp. 303–317, May 1998.
- [225] F. Autrusseau and J. Guédon, "Image watermarking for copyright protection and data hiding via the Mojette Transform," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 378–386.
- [226] M. Barni, F. Bartolini, A. D. Rosa, and A. Piva, "Color image watermarking in the Karhunen-Loeve Transform domain," *Journal of Electronic Imaging*, vol. 11, no. 1, pp. 87–95, Jan. 2002.
- [227] A. Reed and B. Hannigan, "Adaptive color watermarking," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 222–229.

- [228] M. Caramma, R. Lancini, F. Mapelli, and S. Tubaro, "A blind & readable watermarking technique for color images," *Proceedings of the IEEE International Conference on Image Processing 2000*, vol. 1, Thessaloniki, Greece, Sept. 10–13, 2001, pp. 442–445.
- [229] H. S. Malvar and D. A. Florêncio, "An improved spread spectrum technique for robust watermarking," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, May 13–17, 2002, pp. 3301–3304.
- [230] G. L. Guelvouit and S. Pateux, "Wide spread spectrum watermarking with side information and interference cancellation," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, CA, Jan. 21–24, 2003, pp. 278–289.
- [231] R. L. Pickholtz, D. L. Schilling, and L. B. Milstein, "Theory of spread-spectrum communications—a tutorial," *IEEE Transactions on Communications*, vol. COM-30, no. 5, pp. 855–884, May 1982.
- [232] E. H. Dinan and B. Jabbari, "Spreading codes for direct sequence CDMA and wideband CDMA cellular networks," *IEEE Communications Magazine*, vol. 36, no. 9, pp. 48–54, Sept. 1998.
- [233] M. Ejima and A. Miyazaki, "A wavelet-based watermarking for digital images and video," *Proceedings of the IEEE International Conference on Image Processing 2000*, vol. 3, Vancouver, Canada, Sept. 10–13, 2000, pp. 678–681.
- [234] C.-H. Li and S.-S. Wang, "Transform-based watermarking for digital images and video," *Proceedings of the IEEE International Conference on Consumer Electronics*, Los Angeles, CA, June 22–24, 1999, pp. 108–109.
- [235] R. Lancini, F. Mapelli, and S. Tubaro, "A robust video watermarking technique in the spatial domain," *Proceedings of the IEEE Region-8 International Symposium on Video / Image Processing and Multimedia Communications*, Zadar, Croatia, June 16–19, 2002, pp. 251–256.
- [236] K. Su, D. Kundur, and D. Hatzinakos, "A novel approach to collusion-resistant video watermarking," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, San Jose, CA, Jan. 21–24, 2002, pp. 491–502.
- [237] G. D. Haan and E. B. Bellers, "Delacing—an overview," *Proceedings of the IEEE*, vol. 86, no. 9, pp. 1839–1857, Sept. 1998.
- [238] M. D. Swanson, B. Zhu, and A. H. Tewfik, "Multiresolution scene-based video watermarking using perceptual models," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 540–550, May 1998.
- [239] F. Deguillaume, G. Csurka, J. O'Ruanaidh, and T. Pun, "Robust 3D DFT video watermarking," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents I*, vol. 3657, San Jose, CA, Jan. 25–27, 1999, pp. 113–124.
- [240] C. V. Serdean, M. A. Ambroze, M. Tomlinson, and J. G. Wade, "DWT-based high-capacity blind video watermarking, invariant to geometrical attacks," *IEE Proceedings of Vision, Image, and Signal Processing*, vol. 150, no. 1, pp. 51–58, Feb. 2003.

- [241] J. H. Lim, D. J. Kim, H. T. Kim, and C. S. Won, "Digital video watermarking using 3D-DCT and intra-cubic correlation," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents III*, vol. 4314, San Jose, CA, Jan. 22–25, 2001, pp. 64–72.
- [242] F. Hartung and B. Girod, "Watermarking of uncompressed and compressed video," *Signal Processing*, vol. 66, no. 3, pp. 283–301, May 1998.
- [243] A. M. Alattar, E. T. Lin, and M. U. Celik, "Digital watermarking of low bit-rate advanced simple profile MPEG-4 compressed video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 787–800, Aug. 2003.
- [244] G. C. Langelaar, R. L. Lagendijk, and J. Biemond, "Real-time labeling of MPEG-2 compressed video," *Journal of Visual Communication and Image Representation*, vol. 9, no. 4, pp. 256–270, Dec. 1998.
- [245] J. Zhang, J. Li, and L. Zhang, "Video watermark technique in motion vector," *Proceedings of the XIV Brazilian Symposium on Computer Graphics and Image Processing*, Florianopolis, Brazil, Oct. 15–18, 2001, pp. 179–182.
- [246] M. Kutter, F. Jordan, and T. Ebrahimi, "Proposal of a watermarking technique to hide/retrieve copyright data in video," ISO/IEC JTC1/SC29/WG11, Stockholm, Sweden, Tech. Rep. M2281, July 1997.
- [247] P. Bas and B. Macq, "A new video-object watermarking scheme robust to object manipulation," *Proceedings of the IEEE International Conference on Image Processing 2001*, vol. 2, Thessaloniki, Greece, Oct. 7–10, 2001, pp. 526–529.
- [248] M. Barni, F. Bartolini, V. Cappellini, and N. Checcacci, "Object watermarking for MPEG-4 video streams copyright protection," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 24–26, 2000, pp. 465–476.
- [249] A. Piva, R. Caldelli, and A. D. Rosa, "A DWT-based object watermarking system for MPEG-4 video streams," *Proceedings of the IEEE International Conference on Image Processing 2000*, vol. 3, Vancouver, Canada, Sept. 10–13, 2000, pp. 5–8.
- [250] X. Wu, W. Zhu, Z. Xiong, and Y.-Q. Zhang, "Object-based multiresolution watermarking of images and video," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, Geneva, Switzerland, May 28–31, 2000, pp. 212–215.
- [251] R. Dugad and N. Ahuja, "A scheme for joint watermarking and compression of video," *Proceedings of the IEEE International Conference on Image Processing 2000*, vol. 2, Vancouver, Canada, Sept. 10–13, 2000, pp. 80–83.
- [252] I. Setyawan and R. L. Lagendijk, "Low bit-rate video watermarking using temporally extended Differential Energy Watermarking (DEW) algorithm," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents III*, vol. 4314, San Jose, CA, Jan. 22–25, 2001, pp. 73–84.

- [253] J. Dittmann, M. Steinebach, I. Rimac, S. Fischer, and R. Steinmetz, "Combined video and audio watermarking: Embedding content information in multimedia data," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 24–26, 2000, pp. 455–464.
- [254] D. Delannay and B. Macq, "Classification of watermarking schemes robust against loss of synchronizaton," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan. 19–22, 2004, pp. 581–591.
- [255] A. van Leest, J. Haitzma, and T. Kalker, "On digital cinema and watermarking," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, CA, Jan. 21–24, 2003, pp. 526–535.
- [256] N. V. Boulgouris, F. D. Koravos, and M. G. Strintzis, "Self-synchronizing watermark detection for MPEG-4 objects," *Proceedings of the 8th IEEE International Conference on Electronics, Circuits, and Systems 2001*, vol. 3, Oct. 2–5, 2001, pp. 1371–1374.
- [257] E. W. Weisstein, *CRC Concise Encyclopedia of Mathematics*. Boca Raton, FL: Chapman & Hall/CRC, 1999, pp. 921–923; 1848.
- [258] D. Zheng, J. Zhao, and A. E. Saddik, "RST-Invariant digital image watermarking based on log-polar mapping and phase correlation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 753–765, Aug. 2003.
- [259] S. Pereira, J. J. K. Ó Ruanaidh, F. Deguillaume, G. Csurka, and T. Pun, "Template based recovery of Fourier-based watermarks using log-polar and log-log maps," *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, vol. 1, Florence, Italy, June 7–11, 1999, pp. 870–874.
- [260] A. M. Alattar and J. Meyer, "Watermark re-synchronization using log-polar mapping of image autocorrelation," *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 2, Bangkok, Thailand, May 25–28, 2003, pp. 928–931.
- [261] I. Setyawan, G. Kakes, and R. L. Lagendijk, "Synchronization-insensitive video watermarking using structured noise pattern," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 520–530.
- [262] J. Lichtenauer, I. Setyawan, T. Kalker, and R. Lagendijk, "Exhaustive geometrical search and the false positive watermark detection probability," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, CA, Jan. 21–24, 2003, pp. 203–214.
- [263] X. Niu, M. Schmucker, and C. Busch, "Video watermarking resisting to rotation, scale, and translation," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 512–519.
- [264] A. Herrigel, S. Voloshynovskiy, and Y. Rytsar, "The watermark template attack," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents III*, vol. 4314, San Jose, CA, Jan.22–25, 2001, pp. 394–405.

- [265] E. T. Lin and E. J. Delp, "Spatial synchronization using watermark key structure," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan. 19–22, 2004, pp. 536–547.
- [266] F. Deguillaume, S. Voloshynovskiy, and T. Pun, "A method for the estimation and recovering from general affine transforms in digital watermarking applications," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 313–322.
- [267] M. Kutter, "Watermark resisting to translation, rotation, and scaling," *Proceedings of the SPIE: Multimedia Systems and Applications*, A. G. Tescher, B. Vasudev, J. V. Michael Bove, and B. Derryberry, Eds., vol. 3528, Jan. 1999, pp. 423–431.
- [268] D. Delannay and B. Macq, "Generalized 2-D cyclic patterns for secret watermark generation," *Proceedings of the IEEE International Conference on Image Processing 2000*, vol. 2, Vancouver, Canada, Oct. 10–13, 2000, pp. 77–79.
- [269] C.-H. Lee, H.-K. Lee, and Y. Suh, "Autocorrelation function based watermarking with side information," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, CA, Jan. 21–24, 2003, pp. 349–358.
- [270] J. Lichtenauer, I. Setyawan, and R. Lagendijk, "Hiding correlation-based watermark templates using secret modulation," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan. 19–22, 2004, pp. 501–512.
- [271] P. Bas, J.-M. Chassery, and B. Macq, "Geometrically invariant watermarking using feature points," *IEEE Transactions on Image Processing*, vol. 11, no. 9, pp. 1014–1028, Sept. 2002.
- [272] J. S. Seo and C. D. Yoo, "Image watermarking based on scale-space representation," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan. 19–22, 2004, pp. 560–570.
- [273] C.-W. Tang and H.-M. Hang, "A feature-based robust digital image watermarking scheme," *IEEE Transactions on Signal Processing*, vol. 51, no. 4, pp. 950–959, Apr. 2003.
- [274] A. Nikolaidis and I. Pitas, "Robust watermarking of facial images based on salient geometric pattern matching," *IEEE Transactions on Multimedia*, vol. 2, no. 3, pp. 172–184, Sept. 2000.
- [275] C.-P. Wu, P.-C. Su, and C.-C. J. Kuo, "Robust and efficient digital audio watermarking using audio content analysis," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 24–26, 2000, pp. 382–392.
- [276] E. Hauer and S. Thiemert, "Synchronization techniques to detect MPEG video frames for watermark retrieval," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan. 19–22, 2004, pp. 315–324.

- [277] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. Cambridge, United Kingdom: Cambridge University Press, 1992, pp. 274–328.
- [278] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 1990.
- [279] D. E. Knuth, *Seminumerical Algorithms*, 3rd ed., ser. The Art of Computer Programming. Reading, MA: Addison-Wesley, 1997, vol. 2.
- [280] M. Matsumoto and T. Nishimura, “Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator,” *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30, Jan. 1998.
- [281] G. Marsaglia. (1996) DIEHARD: A battery of tests of randomness (source code). <http://stat.fsu.edu/~geo/diehard.html>
- [282] J. F. Wakerly, *Digital Design Principles and Practices*. Englewood Cliffs, NJ: Prentice Hall, 1990.
- [283] C. Fischer and R. LeBlanc Jr, *Crafting a Compiler With C*. Redwood City, CA: Benjamin/Cummings, 1991.
- [284] M. D. Davis, R. Sigal, and E. J. Weyuker, *Computability, Complexity, and Languages*, 2nd ed. Boston, MA: Academic Press, 1994.
- [285] M. R. Garey and D. S. Johnson, *Computers and Intractability A Guide to the Theory of NP-Completeness*. New York, NY: W. H. Freeman and Company, 1979.
- [286] S. Cass, “Mind games,” *IEEE Spectrum*, pp. 40–44, Dec. 2002.
- [287] J. Fridrich and M. Goljan, “Steganalysis of digital images — State of the art,” *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 1–13.
- [288] —, “On estimation of secret message length in LSB steganography in spatial domain,” *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan. 19–22, 2004, pp. 23–45.
- [289] J. Fridrich, M. Goljan, and D. Soukal, “Higher-order statistical steganalysis of palette images,” *Proceedings of the SPIE Security and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, CA, Jan. 21–24, 2003, pp. 179–190.
- [290] S. Dumitrescu, X. Wu, and Z. Wang, “Detection of LSB steganography via sample pair analysis,” *IEEE Transactions on Signal Processing*, vol. 51, no. 7, pp. 1995–2007, July 2003.
- [291] E. T. Lin and E. J. Delp, “Temporal synchronization in video watermarking,” *IEEE Transactions on Signal Processing: Supplement on Secure Media*, vol. 52, no. 10, pp. 3007–3022, Oct. 2004.

- [292] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, "Cryptographic hash functions: A survey," Department of Computer Science, University of Wollongong, Tech. Rep. 95-09, July 1995.
- [293] D. Delannay and B. Macq, "A method for hiding synchronization marks in scale and rotation resilient watermarking schemes," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 548–554.
- [294] *Secure Hash Standard*, National Institute of Standards and Technology Std. 180-1, Apr.17, 1995.
- [295] I. L. MacDonald and W. Zucchini, *Hidden Markov and Other Models for Discrete-Valued Time Series*. New York, NY: Chapman & Hall, 1997.
- [296] J. Fridrich, "Visual hash for oblivious watermarking," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, Jan. 24–26, 2000, pp. 286–294.
- [297] M. Álvarez Rodríguez and F. Pérez-González, "Analysis of pilot-based synchronization algorithms for watermarking of still images," *Signal Processing: Image Communication*, vol. 17, no. 8, pp. 611–633, Sept. 2002.
- [298] J. Fridrich, M. Goljan, and A. C. Baldoza, "New fragile authentication watermark for images," *Proceedings of the IEEE International Conference on Image Processing 2000*, vol. 1, Vancouver, Canada, Sept. 10–13, 2000, pp. 446–449.
- [299] M. Yeung and F. Mintzer, "Invisible watermarking for image verification," *Journal of Electronic Imaging*, vol. 7, no. 3, pp. 578–591, July 1998.
- [300] C.-S. Lu and H.-Y. M. Liao, "Structural digital signature for image authentication: An incidental distortion resistant scheme," *IEEE Transactions on Multimedia*, vol. 5, no. 2, pp. 161–173, June 2003.
- [301] C.-Y. Lin and S.-F. Chang, "A robust image authentication method distinguishing JPEG compression from malicious manipulation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 2, pp. 153–168, Feb. 2001.
- [302] L. M. Marvel, G. W. Hartwig, Jr., and C. Boncelet, Jr., "Compression-compatible fragile and semi-fragile tamper detection," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents II*, San Jose, CA, Jan. 24–26, 2000, pp. 131–139.
- [303] J. Dittmann, "Content-fragile watermarking for image authentication," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents III*, vol. 4314, San Jose, CA, Jan. 22–25, 2001, pp. 175–184.
- [304] H. S. Bassali, J. Chhugani, S. Agarwal, A. Aggarwal, and P. Dubey, "Compression tolerant watermarking for image verification," *Proceedings of the IEEE International Conference on Image Processing 2000*, vol. 1, Vancouver, Canada, Sept. 10–13, 2000, pp. 430–433.



- [305] C. Fei, D. Kundur, and R. Kwong, "Analysis and design of authentication watermarking," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan.19–22, 2004, pp. 760–771.
- [306] M. P. Queluz, "Spatial watermark for image content authentication," *Journal of Electronic Imaging*, vol. 11, no. 2, pp. 275–285, Apr. 2002.
- [307] J. Fridrich and M. Goljan, "Images with self-correcting capabilities," *Proceedings of the IEEE International Conference on Image Processing 1999*, Kobe, Japan, Oct. 1999.
- [308] J. Fridrich, "Image watermarking for tamper detection," *Proceedings of the IEEE International Conference on Image Processing 1998*, vol. 2, Chicago, IL, Oct. 1998, pp. 404–408.
- [309] —, "Security of fragile authentication watermarks with localization," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 691–700.
- [310] M. Holliman and N. Memon, "Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 432–441, Mar. 2000.
- [311] J. Tian, "Wavelet-based reversible watermarking for authentication," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, Jan. 21–24, 2002, pp. 679–690.
- [312] J. Fridrich, M. Goljan, and R. Du, "Invertible authentication," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents III*, vol. 4314, San Jose, CA, Jan. 22–25, 2001, pp. 197–208.
- [313] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Localized lossless authentication watermark (LAW)," *Proceedings of the SPIE Security and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, CA, Jan. 21–23, 2003, pp. 689–698.
- [314] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Transactions on Image Processing*, vol. 13, no. 8, pp. 1147–1156, Aug. 2004.
- [315] S. Katzenbeisser and J. Dittmann, "Malicious attacks on media authentication schemes based on invertible watermarks," *Proceedings of the SPIE Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, Jan. 19–22, 2004, pp. 838–847.
- [316] F. Mintzer and G. Braudaway, "If one watermark is good, are more better?" *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing 1999*, vol. 4, Phoenix, AZ, May 1999.
- [317] Ö. Ekici, B. Sankur, B. Coşkun, U. Naci, and M. Akcay, "Comparative evaluation of semifragile watermarking algorithms," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 209–216, Jan. 2004.

VITA

## VITA

Eugene Lin was born in Stillwater, Oklahoma. He received the B.S. Computer and Electrical Engineering degree in 1994, the M.S. Electrical Engineering degree in 1996, and the Ph.D. Computer Engineering in 2005 at Purdue University.

He was an intern at Lucent Technologies in the summer of 2000. In 2001 and 2002, he was a summer intern at Digimarc Corporation.

He is a student member of the IEEE and a member of Eta Kappa Nu. His research interests include video watermarking and steganography, as well as video coding and image processing.

**Journal Publications:**

- A. M. Alattar, E. T. Lin, and M. U. Celik, “Digital Watermarking of Low Bit-Rate Advanced Simple Profile MPEG-4 Compressed Video,” *IEEE Transactions on Circuits and Systems for Video Technology: Special Issue on Authentication, Copyright Protection, and Information Hiding*, vol. 13, no. 8, pp. 787–800, August 2003.
- E. T. Lin, C. I. Podilchuk, T. Kalker, and E. J. Delp, “Streaming video and rate scalable compression: What are the challenges for watermarking?,” *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 198–205, January 2004.
- E. T. Lin and E. J. Delp, “Temporal Synchronization in Video Watermarking,” *IEEE Transactions on Signal Processing: Supplement on Secure Media*, vol. 52, no. 10, pp. 3007–3022, October 2004.
- E. T. Lin, A. M. Eskicioglu, R. L. Lagendijk, and E. J. Delp, “Advances in Digital Video Content Protection,” *Proceedings of the IEEE: Special Issue on*

*Advances in Video Coding and Delivery*, vol. 93, no. 1, pp. 171–183, January 2005.

### Conference Papers:

- E. T. Lin and E. J. Delp, “A Review of Data Hiding in Digital Images,” *Proceedings of the Image Processing, Image Quality, Image Capture Systems Conference (PICS '99)*, Savannah, GA, April 25–28, 1999, pp. 274–278.
- E. T. Lin and E. J. Delp, “A Review of Fragile Image Watermarks,” *Proceedings of the Multimedia and Security Workshop (ACM Multimedia '99) Multimedia Contents*, Orlando, FL, October 1999, pp. 25–29.
- E. T. Lin, C. I. Podilchuk, and E. J. Delp, “Detection of Image Alterations Using Semi-Fragile Watermarks,” *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents II*, vol. 3971, San Jose, CA, January 23–28, 2000, pp. 152–163.
- E. T. Lin, C. I. Podilchuk, A. Jacquin, and E. J. Delp, “A Hybrid Embedded Video Codec Using Base Layer Information for Enhancement Layer Coding,” *Proceedings of the International Conference on Image Processing 2001*, Thessaloniki, Greece, October 7–10, 2001.
- E. T. Lin, C. I. Podilchuk, T. Kalker, and E. J. Delp, “Streaming Video and Rate Scalable Compression: What Are the Challenges for Watermarking?,” *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents III*, vol. 4314, San Jose, CA, January 22–25, 2001, pp. 116–127.
- E. T. Lin and E. J. Delp, “Temporal Synchronization in Video Watermarking,” *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents IV*, vol. 4675, San Jose, CA, January 20–25, 2002, pp. 478–490.

- E. T. Lin and E. J. Delp, “Temporal Synchronization in Video Watermarking—Further Studies,” *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, CA, January 20–24, 2003, pp. 493–504.
- A. Alattar, M. U. Celik, and E. T. Lin, “Evaluation of Watermarking Low Bit-Rate MPEG-4 Bit Streams,” *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents V*, vol. 5020, Santa Clara, CA, January 20–24, 2003, pp. 440–451.
- A. M. Alattar, E. T. Lin, and M. U. Celik, “Watermarking Low Bit-rate Advanced Simple Profile MPEG-4 Bitstreams,” *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 2003*, Hong Kong, April 6–10, 2003, vol. 3, pp. 513–516.
- O. Guitart Pla, E. T. Lin, and E. J. Delp, “A Wavelet Watermarking Algorithm Based on a Tree Structure,” *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, January 19–22, 2004, pp. 571–580.
- E. T. Lin and E. J. Delp, “Spatial Synchronization Using Watermark Key Structure,” *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306, San Jose, CA, January 19–22, 2004, pp. 536–547.
- A. Lang, J. Dittmann, E. T. Lin, and E. J. Delp, “Application-Oriented Audio Watermark Benchmark Service,” to appear in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents VII*, San Jose, CA, January 17 - 20, 2005.
- H. C. Kim, E. T. Lin, and E. J. Delp, “Further Progress in Watermarking Evaluation Testbed (WET),” to appear in *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents VII*, San Jose, CA, January 17 - 20, 2005.

- E. T. Lin, Y. Liu, and E. J. Delp, “Detection of Mass Tumors in Mammograms using SVD Subspace Analysis,” to appear in *Proceedings of the SPIE International Conference on Computational Imaging III*, San Jose, CA, January 17–18, 2005.